

## **Part III**

# **Collaborative and Evolutionary Intelligent Systems**

The concepts of **collaboration** and **evolution** lie at the heart of intelligent multi-agent systems (MAS). Inspired by biological ecosystems and human societal dynamics, these systems leverage collective intelligence to solve complex challenges that exceed the capabilities of individual agents [914]. Human societies exemplify how cooperation, specialization, and distributed decision-making significantly enhance collective problem-solving effectiveness. Similarly, MAS adopts these strategies, integrating specialized agents to address intricate tasks collaboratively. The foundational principle of collective intelligence – the “Wisdom of Crowds” by [915] – suggests diverse, independent agents often yield superior decisions compared to solitary experts, directly underpinning the design philosophy of MAS. Cognitive theories, such as Minsky’s society of mind [17] and the theory of mind [916, 917], further reinforce this paradigm by proposing that intelligence emerges from structured interactions among specialized units.

Recently, advancements in large language models (LLMs) have introduced new possibilities for collaborative and evolutionary multi-agent systems (LLM-MAS). Benefiting from powerful reasoning, planning, and decision-making capabilities, these models enable the creation of sophisticated MAS architectures mirroring the cooperative and adaptive characteristics found in human societies. Agents within LLM-MAS often assume distinct identities and roles, reflecting human-like division of labor and specialized collaboration. By embracing structured communication, dynamic knowledge sharing, and coordinated decision-making, these systems emulate human social dynamics to achieve common goals. Moreover, LLM-MAS is inherently evolutionary; agents continuously adapt and improve through interactions, feedback, and iterative learning, resulting in enhanced system performance over time. **Roadmap** In this chapter, we systematically survey the emerging field of LLM-based multi-agent systems, focusing specifically on their collaborative mechanisms and evolutionary capabilities. We first examine how distinct system objectives shape agent roles, behavior patterns, and collaborative strategies in Chapter 13. Next, in Chapter 14, we analyze various communication structures, including interaction protocols that facilitate effective agent-agent and human-agent communication. Additionally, we explore collaborative decision-making methodologies and how agents leverage their unique expertise and perspectives in Chapter 15, and discuss the collective intelligence and evolution mechanism in Chapter 16. Finally, in Chapter 17, we discuss evolutionary processes, highlighting adaptive learning methods, continuous knowledge sharing, and mechanisms for iterative improvement that collectively enhance MAS performance. Through this comprehensive survey, we identify current achievements, discuss existing challenges, and highlight promising research directions for collaborative and evolutionary intelligent systems.

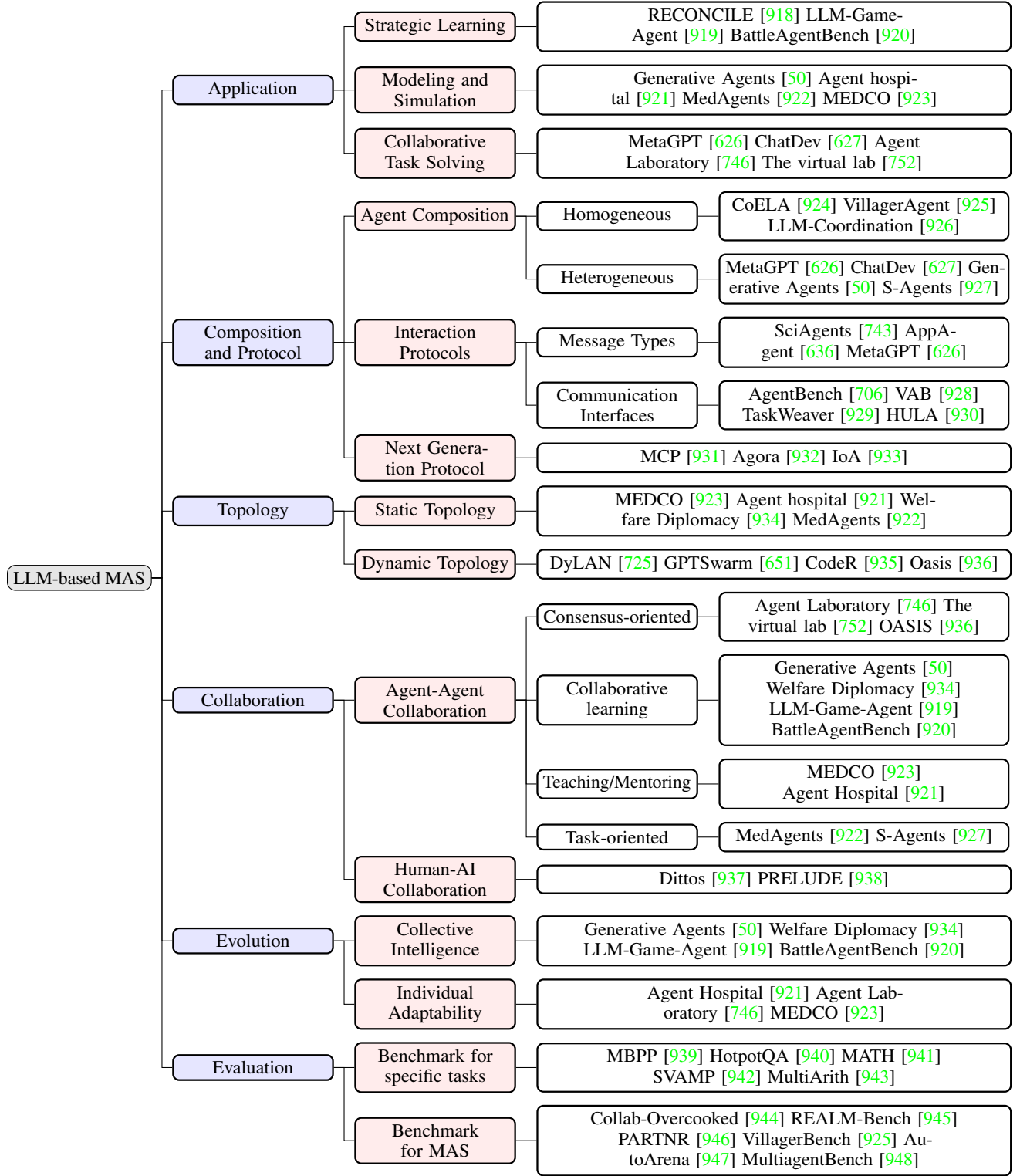


Figure 12.3: Taxonomy of LLM-based Multi-Agent Systems.

## Chapter 13

# Design of Multi-Agent Systems

In the context of LLM-based multi-agent systems (LLM-MAS), *collaboration goals* and *collaboration norms* serve as foundational elements that shape system behavior, interaction patterns, and overall effectiveness. Collaboration goals specify the explicit objectives agents aim to achieve – whether individually, collectively, or competitively – while collaboration norms define the rules, constraints, and conventions that govern agent interactions within the system. Together, these components establish a robust framework guiding effective communication, coordination, and cooperation among agents.

This section categorizes LLM-MAS into three broad classes based on distinct combinations of collaboration goals and norms: *strategic learning*, *modeling and simulation*, and *collaborative task solving*. Although not exhaustive, these categories cover a wide spectrum of LLM-MAS designs and clearly reflect how system objectives shape agent interactions and outcomes.

- **Strategic Learning** systems embed agents within a game-theoretic context, where agents pursue individual or partially conflicting goals. The interactions can be cooperative, competitive, or mixed, guided explicitly by predefined game rules and interaction norms. This setting often aligns with non-cooperative (strategic) and cooperative concepts in traditional game theory. Please refer to Section 13.1 for details.
- **Modeling and Simulation** contexts focus on agents acting independently, driven by diverse environmental or social factors. Here, interactions emerge organically without necessarily converging on common goals, reflecting the complex dynamics seen in large-scale social or economic simulations. Please refer to Section 13.2 for details.
- **Collaborative Task Solving** emphasizes systematic cooperation among agents to achieve explicitly shared objectives. Agents typically adopt structured workflows, clear role definitions, and highly predefined collaboration norms to synchronize their actions toward collective goals. Please refer to Section 13.3 for details.

In the remainder of this chapter, we elaborate on each category, examining how LLMs enable, influence, and enhance agent behaviors, interactions, and collective intelligence within our scope.

In the following, we examine these categories in detail, highlighting how each leverages the capabilities of large language models to shape agent behaviors and interactions.

### 13.1 Strategic Learning: Cooperation vs. Competition

**Strategic learning** refers to agents’ capabilities to dynamically anticipate, interpret, and influence the actions of other agents within game-theoretic settings—whether competitive, cooperative, or mixed [949]. Agents iteratively adjust their strategies based on new information, commonly modeled using foundational concepts such as Nash equilibria [950], Bayesian games [951, 914, 952], or repeated interactions [953, 954]. With LLMs enabling nuanced linguistic reasoning, strategic learning increasingly integrates “soft” signals – including dialogue, persuasion, and implicit negotiation – thus enriching traditional game-theoretic reasoning frameworks [952, 955, 956, 957].

In economic applications, multi-agent strategic simulations provide valuable insights into market behaviors and negotiation tactics, highlighting both competitive and cooperative dynamics. For example, [958] and [951] demonstrate how LLM-empowered agents can simulate hiring processes, exhibit rational decision-making in controlled economic

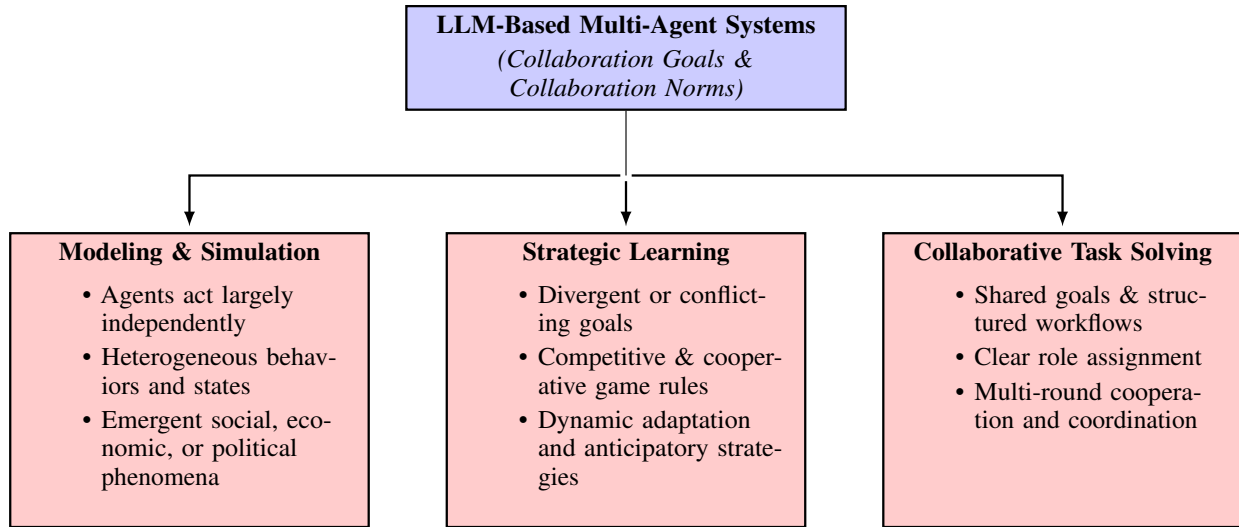


Figure 13.1: An overview of three major collaboration types in LLM-based MAS: *Modeling & Simulation*, *Strategic Learning*, and *Collaborative Task Solving*. Each category is distinguished by how agents’ goals and norms are set (independent vs. divergent vs. shared) and how they coordinate.

experiments, and even forecast stock movements. [959] introduces a GPT-4-based competitive environment to illustrate how restaurant and customer agents compete to optimize profits and satisfaction, showcasing realistic bidding and pricing strategies. Meanwhile, [960] investigate Buyer–Seller bargaining in LLM-based negotiations, while [961] use ultimatum game simulations to illuminate policymaking decisions grounded in human-like strategic behavior.

Beyond conventional markets, strategic learning applies broadly wherever resource allocation, alliances, or competitive-cooperative trade-offs are present. Examples include multi-commodity competitions [962, 959], in which agents strategically negotiate terms to maximize individual benefits, or sustainability-focused contexts where agents coordinate resource consumption [963]. In gaming, social deduction games such as Werewolf, Chameleon, Avalon, and Jubensha require agents to manage the complex interplay between deception and collaboration [964, 965, 966, 153, 919, 967, 968, 969, 970]. Studies by [971, 965] highlight LLM-based agents that excel at orchestrating subtle deceit and collaboration, while [967, 972, 968, 969] emphasize adaptive, multi-round strategy in Avalon. [970] further pushes this boundary by showcasing autonomous, multi-agent interactions in the *Jubensha* murder mystery genre, re-creating complex narratives. Similarly, diplomatic simulations ([973] and [974]) employ LLM-based agents to emulate sophisticated geopolitical negotiation and alliance formation dynamics at global scales.

**Summary** A key advantage of LLM-driven strategic learning lies in effectively combining rigorous game-theoretic logic with natural language reasoning. This fusion enables agents to interpret sophisticated instructions, engage in persuasive dialogue, and adapt more flexibly to novel or unstructured settings. Consequently, LLM-based strategic agents hold significant promise for accurately modeling complex real-world interactions – spanning economic competition, social negotiation, and geopolitical strategy – far more effectively than conventional rule-based or numeric-only approaches.

## 13.2 Modeling Real-World Dynamics

**Modeling and simulation** represents another crucial area of application for LLM-based multi-agent systems (LLM-MAS), aiming to replicate complex social, economic, and political phenomena at scale. By utilizing LLMs’ sophisticated language understanding and contextual reasoning, these simulations can feature highly heterogeneous agents whose evolving behaviors mirror real-world dynamism. Unlike strategic learning environments that emphasize explicit competitive or cooperative goals, agents in modeling and simulation scenarios operate independently, guided by their domain-specific roles, preferences, and interactions with the simulated environment [975].

In healthcare, for example, [921] introduces *Agent Hospital*, where LLM-powered doctor agents iteratively refine treatment strategies through realistic interactions with virtual patients. This enables researchers to test management protocols, training paradigms, and “what-if” scenarios in a controlled yet realistic setting. Similarly, in economic contexts, [976] present *EconAgents*, leveraging LLM-driven agents to realistically model individual-level behaviors such as employment decisions, consumption patterns, and savings strategies. These agents facilitate expressive macroe-

conomic simulations, surpassing traditional numeric or strictly rule-based methods in adaptability and realism [977]. In addition, political science applications also benefit from this approach. For example, [978] and [977] successfully simulate election processes and policymaking dynamics, revealing how public discourse, candidate strategies, and voter interactions shape real-world political outcomes.

Beyond economics and politics, LLM-based simulation accommodates a variety of social and cultural phenomena. For example, [979] and [255] use simulations of linguistic and emotional propagation in social networks to investigate how opinions, beliefs, or sentiment clusters form online. Research by [980] explores how opinion dynamics evolve under various topological and interaction patterns, while [981] examines the conditions under which fake news spreads or stalls in heterogeneous agent populations. Large-scale simulation platforms such as GenSim [982] and OASIS [936] push the boundary further by scaling to tens of thousands or even millions of user agents, thus enabling the study of emergent group behaviors and systemic effects—such as viral information diffusion, echo-chamber formation, or group polarization—under realistic constraints.

**Summary** The strength of LLM-based simulation lies in capturing both the structural dynamics (e.g., network topology or institutional rules) and the cognitive or linguistic nuances that drive real-world behavior. By embedding language-based reasoning into agent models, researchers can examine complex social processes—like persuasion, framing, or cultural transmission—that would be difficult to capture through purely numeric or rule-based approaches.

### 13.3 Collaborative Task Solving with Workflow Generation

**Collaborative task solving** orchestrates multiple agents toward a clearly defined objective through structured workflows. In contrast to strategic learning (which may involve competing interests) or open-ended modeling and simulation (where agents act independently), collaborative agents function as part of a unified problem-solving pipeline. Agents typically follow clearly defined roles (e.g., “Planner”, “Implementer”, or “Evaluator”) and stage-based processes to ensure efficient and accurate task completion.

Systems such as MetaGPT [626], CAMEL [848], Communicative Agents [983], and frameworks described in [924] exemplify how clearly defined roles, responsibilities, and decision flows allow LLM-based agents to coordinate effectively. A typical workflow might involve one agent analyzing a problem statement, another proposing a solution outline, a third implementing partial solutions, and a fourth verifying correctness. Communication among these agents is often carried out through iterative rounds of natural language “dialogue”, leveraging the inherent language-generation strengths of LLMs. This structured approach also proves beneficial for scaling to more ambitious projects, as sub-tasks can be delegated to specialized agents with domain-specific prompts or training.

Recently, collaborative task-solving systems have been explored extensively in software development scenarios (e.g., multi-agent coding, debugging, and testing). However, scientific discovery represents a particularly prominent and compelling application. For example, the *Agent Laboratory* [746] employs agents in structured scientific workflows: proposing hypotheses, designing experiments, analyzing results, and refining subsequent inquiries, which effectively mirrors the iterative nature of the scientific investigation. Similar multi-agent designs can be adapted to tasks such as literature review, policy drafting, or large-scale data analysis, using well-defined protocols to maintain coherence and avoid duplication of effort.

**Summary** Compared to other LLM-based multi-agent paradigms, collaborative task-solving inherently prioritizes clarity and predictability: Each agent’s role and objective are predefined, limiting emergent or chaotic behaviors. This structure is particularly advantageous in domains requiring precision, accountability, or sequential decision-making. At the same time, research is ongoing to strike the right balance between structure and flexibility, which ensures that agents have enough autonomy to creatively contribute solutions while adhering to a shared workflow that ultimately guarantees reliable, high-quality task completion.

**Discussion** The aforementioned three dimensions—*strategic learning*, *modeling and simulation*, and *collaborative task solving*—reflect the breadth of LLM-based multi-agent systems. Each category addresses distinct research questions and real-world applications, leveraging language-based reasoning to tackle challenges that extend beyond the capabilities of conventional, purely numeric, or rule-driven agent designs.

### 13.4 Composing AI Agent Teams

In MAS, agents are the core units that interact within the system and are critical to its functionality. These agents can be categorized as either homogeneous or heterogeneous, depending on whether they share identical or differing personas, capabilities, and action spaces.

**Homogeneous** Homogeneous agents that share identical capabilities, action spaces, and observation spaces. Compared to single-agent systems, the primary advantage lies in task parallelization, allowing multiple agents to handle different parts of a task simultaneously and improve overall efficiency. They are often used in simpler, coordinated tasks where uniformity across agents can drive improved performance.

Several studies have applied homogeneous agents to simulate teamwork in games like Overcooked and Minecraft, as well as real-world tasks such as household labor division. [924] proposed a cognitive-inspired modular framework that enables LLM-based agents to communicate through natural language to perform labor division, request assistance from one another, and collaboratively complete object transportation tasks. [984] introduced prompt-based organizational structures into the framework, reducing communication costs between agents and improving team efficiency in household tasks such as preparing afternoon tea, washing dishes, and preparing a meal. Furthermore, several studies [926, 925] have employed multiple LLM-based agents in popular games such as Overcooked and Minecraft to experiment with their ability to cooperate and complete tasks. According to the game settings, these agents are also homogeneous.

**Heterogeneous** Agent diversity plays a crucial role in improving collaboration outcomes. Research shows that heterogeneity among agents can enhance problem-solving capabilities, as diverse agents bring varied perspectives and skills to the task at hand [985, 986]. Heterogeneity contributes to richer problem-solving strategies and improves overall collaboration in MAS. The heterogeneous characteristics of agents can be reflected in the following dimensions: personas-level heterogeneity, observation-space heterogeneity, and action-space heterogeneity. Note that these heterogeneities are not mutually exclusive—a heterogeneous agent may exhibit one or more of these characteristics.

- *Personas-level heterogeneity.* Refers to diversity in agent profiles, which influences how agents approach problem-solving and interact with one another. Most current LLM-based heterogeneous multi-agent systems fall into this category [987, 627, 50, 970]. For example, in software development, agents may take on personas such as programmers, product managers, or testers. In medical diagnostics, agents may represent cardiologists, oncologists, or paediatricians, each with distinct areas of expertise. The distinct perspectives and expertise of each persona contribute to more robust decision-making. While these heterogeneous agents may share the same action space—such as writing documents [626] (e.g., code, requirement reports, or test reports) or providing diagnostic advice [922]—their personas influence the outcomes of these actions, where role-specific enhancements within multi-agent architectures have shown to significantly streamline and optimize task execution. For instance, a product manager performing the action of writing a document would produce a requirements report, whereas a programmer performing the same action would produce software implementation code [626]. This diversity leads to better decision-making and innovation, especially in complex, multidisciplinary tasks.
- *Observation-space heterogeneity.* In MAS, the ability of agents to perceive and interpret their environment can vary. Observation-space heterogeneity refers to these differences in what agents can observe or perceive within their environment. For example, in the game Werewolf, some agents, like werewolves, can see the identities of their teammates, and the seer can obtain the identity of a designated player, while others, like villagers, cannot see the true identity of any player [971]. Similarly, in the Avalon game, different roles have distinct observation spaces [919, 972], thus influencing the strategies and communications of the players. In these settings, each agent’s perceptual ability or observation space is directly linked to their role in the system. In a multi-agent system, this variation in what agents can observe often influences their decision-making, communication, and coordination with other agents.
- *Action-space heterogeneity.* On the other hand, this refers to fundamental differences in the actions agents can perform due to physical or functional constraints. This is particularly relevant in both virtual and physical environments where agents may have different capabilities based on their design or purpose. In the virtual environments of games like Werewolf [965, 971, 966] and Avalon [919, 967], different roles have distinct abilities or skills [971, 919, 972]. For example, in Werewolf, while werewolves may have the ability to communicate secretly with each other, villagers might be limited to voting or observing only. This dynamic requires agents to collaborate based on their unique capabilities and promotes the learning of strategies such as teamwork, trust, and deception in their interactions. Meanwhile, in robotics, agents may exhibit diverse physical capabilities. For instance, as described in [988], some robots lack mobility and can only manipulate objects, while others are specialized for movement but cannot manipulate objects. In such cases, agents with different action spaces must divide tasks effectively, leveraging their specific abilities to take on the parts of the task they are suited for, ultimately collaborating to complete the overall task. This type of heterogeneity requires agents to collaborate and coordinate their actions efficiently, often dividing tasks based on their individual strengths.

**Homogeneity to Heterogeneous Evolution** In some LLM-based multi-agent systems, agents have the ability to evolve autonomously and continuously adapt through interactions with their environment. Due to the inherent randomness



in both LLM models and the environment, the evolution of these agents often follows different trajectories. This can lead to heterogeneous behaviors emerging over multiple simulations, even when agents initially have homogeneous personas and action spaces. For example, as shown in [989], agents with identical action spaces and personas at the start developed differentiated roles after multiple rounds of interactions with the environment and other agents. Some agents, for instance, specialized in food gathering, while others focused on crafting weapons. Similarly, [990] observed that initially homogeneous agents developed distinct language usage patterns, emotional expressions, and personalities after group interactions. These emergent behaviors demonstrate the possibility of transitions from homogeneous to heterogeneous systems.

## 13.5 Agent Interaction Protocols

In this section, there will initially be classification of typical kinds of messages, providing a clear view regarding the content and exchange modes for agent interactions. Next, agent-environment, agent-agent, and agent-human communications interface designs will be addressed. Architectural issues and protocol specifications for transparent information exchange will also be addressed. Interface standardization will have a special focus, which is essential for providing interoperability, scalability, and efficiency for multi-agent systems. The section will end with unification of communication protocol discussions, where agent-environment or agent-user interacting design principles and requirements are addressed, as well as providing clarity, consistency, and functional coherence for various applications for LLM-based systems.

### 13.5.1 Message Types

**Structured:** Structured messages, either in JSON ([991, 992]), XML ([993, 636]), or as a code ([626, 627, 994]), are a crucial aspect of multi-agent system communication with LLM. The primary advantages of structured messages are their syntactically and semantically defined structure, enabling unambiguous understanding and straightforward parsing. With their lack of ambiguity, they facilitate unerrant information extraction and processing with much less overhead on computation and greater system dependability. For example, JSON and XML can represent specific-task configuration parameters or facilitate data exchange as a machine-readable mode, and messages written as a code can even be executable several times directly, which makes workflow and automation simpler.

Structured messages are particularly well-suited for high-efficiency, deterministic applications. They are useful for sub-task decomposition, sub-task assignment, and coordination among agents for cooperative multi-agent architecture because they explicitly state operational commands. Moreover, as structured messages have a prescribed form, retrieving data as well as storing data is facilitated and system optimization and longitudinal analysis are also feasible.

**Unstructured:** In contrast, unstructured messages, e.g., natural text ([971, 970, 919]), visual data, e.g., images, videos, and audio signals, e.g., speech, ambient sounds ([995, 996, 762]), have higher information density and representational capability. Such modalities are best suited for communication with nuanced and context-dependent information. Images, for instance, communicate spatial relationships, illumination, and facial expressions, and videos communicate dynamic temporally-organized sequences, e.g., state or behavior changes over time. Similarly, audio signals also communicate not just linguistic information but also paralinguistic information, e.g., tone, emotion, and intonation, which are critical for natural and context-aware interactions.

Unstructured messages are well-adapted for ambiguity tasks, as well as for complex, real-world settings. The fact that they can express abstract ideas as well as affective subtlety, or implicit contextual suggestions, makes unstructured messages well-suited for creative, as well as discovery-oriented, problem spaces. Unstructured data's complexity, however, calls for advanced processing techniques, for example, feature extraction based on deep learning, for one to tap into their full potential. Advances with pre-trained LLMs as well as multi-modal large language models have alleviated these complexities to a large extent, enabling novel applications for unstructured communication within multi-agent systems [533, 513, 997].

**Summary:** Unstructured and structured messages have complementary roles for multi-agent communication with LLM-based. While structured messages offer accuracy, consistency, and computation efficiency and are appropriate for operational and deterministic operations, unstructured messages offer rich, contextualized representations enabling agents to negotiate vague, creative, highly dynamic situations. Together, these modes offer a foundation for adaptive, effective multi-agent cooperation.



### 13.5.2 Communication Interface

**Agent-Environment Interface** LLM-based agents will typically have to act on their environment once or several times in order to perform a range of operations. From the agent’s point of view, its output into the environment is something that it would prefer, e.g., a UI click, web request, or a move for a computer graphic’s character. Environments differ with regard to what actions they will accept, and so as not have its actions not get executed, the agent must find out what actions are for a specific environment that it is acting within and perform actions that are for a specific task as well as valid for a specific environment. After the agent outputs its chosen action, the agent will have a return from the environment. It will consist of observations if successful, or a feedback on error if there was one. The agent will have to act on this feedback. There are nowadays various types of environments where an agent can act, e.g., operating systems, computer games, database, and e-commerce websites. To make agent-environment interfaces share a common interface and have agents trained on various LLMs plug into various environments with minimal further adaption, various frameworks have been proposed. These frameworks make for easier tests on agents’ capability on various executable environments [706].

**Agent-Agent Communication** In MAS, communication through natural language is predominant. This is likely because large language models possess strong linguistic capabilities due to pretraining on massive natural language corpora. Another possible reason is that, for many tasks, natural language communication is already sufficient to meet the requirements. Based on the type of information exchanged, multi-agent systems can be categorized as follows: *Natural Language-Based Systems* Among LLM-based multi-agent systems utilizing natural language, text-based communication is the most common [922, 924, 987, 970, 998]. There are also some systems that use voice as the medium of communication [996, 762, 999, 1000]. In these systems, agents engage in behaviors such as discussions, negotiations, persuasion, or critique through natural language to achieve their objectives. *Structured Information-Based Systems* Compared to natural language, structured information has characteristics such as higher consistency, lower parsing complexity, and reduced ambiguity, making it more suitable for efficient and low-cost communication between agents [626]. In some implementations, the information exchanged between agents is structured into distinct components to facilitate easier parsing and utilization by the receiving agent. For instance, the exchanged information might include fields specifying the sender, receiver, message type, and instructions on how the recipient should parse or use the content [929].

**Human-Agent Communication** The purpose of developing multi-agent systems is to expand the boundaries of human capabilities and cognition, ultimately serving human well-being. While in some social simulation multi-agent systems, humans primarily exist as observers [50, 1001], most multi-agent systems allow human participation in various forms. During this participation, humans need to communicate with agents, and this communication can take the form of either natural language or structured information [924, 930]. When human-to-agent communication primarily relies on natural language, a single LLM often acts as a hub to parse human natural language into structured information that agents can process more effectively for subsequent operations. This hub LLM can either exist within the multi-agent system or function independently of it. To save time and enhance communication efficiency, humans can also use structured information to communicate with the multi-agent system through programming or similar methods. By following predefined communication protocols, humans can send messages containing the required data to the multi-agent system. The system will then process the messages and data according to its internal logic and return the results. [931]

### 13.5.3 Next-Generation Communication Protocols

The field of LLM-based agents is still in its infancy. Developers typically design agent architectures and communication mechanisms tailored to specific domains or tasks, including agent-to-environment, agent-to-human, and inter-agent interactions. However, most existing systems lack a unified communication framework, resulting in fragmented, siloed ecosystems. Multi-agent systems, tools, environments, and data sources often operate independently, making it difficult for agents to interoperate or share capabilities. Furthermore, the burden of learning and implementing bespoke protocols falls on humans, and almost all current protocols are manually designed—a labor-intensive process that often lacks semantic flexibility or scalability.

To address these issues, several new agent communication protocols have been proposed, each targeting different aspects of the protocol design stack.

**Internet of Agents (IoA)** [933] introduces an internet-inspired, instant-messaging-like communication architecture that supports dynamic team formation and task-driven collaboration. Agents register with a central coordination server, which handles identity management and discovery. Communication flows are orchestrated using FSM (Finite State Machine)-based dialogue templates. IoA supports multiple message types, including discussion, task assignment, and triggering mechanisms, and provides structured fields for controlling speaker turns, nested group formation, and

maximum dialogue length. This allows agents to select and adapt message formats to match specific coordination phases, offering flexibility within a fixed schema.

**Model Context Protocol (MCP)** [931], developed by Anthropic, focuses on enabling LLM agents to access structured tools and data. It adopts a fully centralized approach based on OAuth identity authentication, and interactions are constrained to JSON-RPC 2.0 messages. While it lacks a meta-protocol layer or semantic negotiation capabilities, its simple and rigid architecture makes it a practical choice for tool use cases with well-defined APIs. However, MCP sacrifices flexibility and extensibility, requiring manual registration of supported functions.

**Agent Network Protocol (ANP)** [1002] aims to achieve full decentralization. Agents identify themselves through W3C-compliant decentralized identifiers (DIDs) and communicate over encrypted peer-to-peer channels. The protocol includes a meta-protocol layer that enables agents to negotiate which application-level protocol to adopt, supporting semantic protocol selection based on agent capabilities. ANP also allows for multi-protocol support at the application layer (e.g., HTTP, JSON-RPC, natural language), providing strong extensibility and decentralization but does not yet explicitly support public protocol reuse.

**Agora** [932] offers a highly flexible and language-driven protocol mechanism. Instead of registering pre-defined APIs, agents can generate and share Protocol Descriptions (PDs), which are free-text descriptions of communication semantics. Using a large language model, agents can dynamically interpret and execute any PD at runtime. This allows protocols to be created, deployed, and used entirely through language, without any manual registration or configuration. Agora avoids centralized registries and supports decentralized protocol sharing: agents may publish or retrieve PDs from peer-distributed repositories to enable cumulative learning and interoperability across systems.

**Summary:** As shown in Table 13.1, next-generation agent communication protocols differ along key dimensions such as identity and security mechanisms, meta-protocol negotiation capabilities, application-layer flexibility, and the degree of centralization. A unified, secure, scalable, and dynamic protocol infrastructure—where agents can negotiate and co-create protocols on the fly—is critical for enabling large-scale, interoperable agent ecosystems. While current frameworks such as MCP, ANP, Agora, and IoA represent early but promising steps, protocol design remains a rapidly evolving frontier in the development of intelligent agent systems.

Table 13.1: Comparison of four agent communication protocols (MCP, ANP, Agora, IoA) across identity, negotiation, and execution layers.

**PD** = Protocol Description; **DID**:Decentralized Identifier; **LLM**:Large Language Model; **FSM**:Finite State Machine.

Layer	MCP	ANP	Agora	IoA
<b>Identity &amp; Security</b>	OAuth-based centralized identity authentication.	DID-based decentralized identity with encrypted channels.	No centralized registration. Identity derived from PD hash.	Agents register with a central server for identity and discovery.
<b>Meta-Protocol Layer</b>	No meta-protocol layer; relies on pre-defined interfaces.	Uses DID document to negotiate and select appropriate protocol via semantics.	LLM interprets PD text to automatically negotiate and deploy communication protocols.	A centralized discovery mechanism combined with FSM-based dialogue flow control.
<b>Application Protocol Layer</b>	Supports only JSON-RPC 2.0.	Supports multiple protocols such as HTTP and natural language.	Allows arbitrary PD-driven protocols with high flexibility.	Task-driven protocol coordination supporting multiple message formats.
<b>Degree of Centralization</b>	Highly centralized architecture.	Fully decentralized.	Decentralized: no registration or fixed ID, with optional peer-to-peer PD sharing.	Highly centralized architecture with a central coordination server.
<b>Protocol Flexibility</b>	Fixed and rigid; hard to adapt beyond JSON-RPC.	Highly flexible with semantic negotiation.	Extremely flexible; any PD can define a new protocol dynamically.	Moderately high flexibility; agents can select and adapt message formats based on task phases and coordination needs.

Table 13.2: Classification framework for LLM-based multi-agent systems, highlighting different aspects of system design, communication, collaboration, and evolution. Below are our abbreviations, for ease of reference:

M&S = Modeling & Simulation, CTS = Collaborative Task Solving, SL = Strategic Learning, S-D = Static-Decentralized, S-L = Static-Layered, Hom = Homogeneous, Het = Heterogeneous, T/M = Teaching/Mentoring, C-O = Consensus-Oriented, T-O = Task-Oriented, CL = Collaborative Learning, Dict = Dictatorial, D-B = Debate-Based, CI = Collective Intelligence, Ind = Individual.

Paper	System Design	Communication			Collaboration		Evolution
	Category	Typology	Interface	Agent Type	Interaction	Decision	Type
Agent Hospital [921]	M&S	S-D	Text	Het	T/M, C-O	Dict	Ind
Welfare Diplomacy [934]	M&S	S-L	Code, JSON, Text	Hom	CL	Voting	CI
MEDCO[923]	M&S	S-L	Text	Het	T/M, C-O	Dict	Ind
MedAgents[922]	M&S	S-L	Text	Hom	T-O	Dict	CI
Generative Agents [50]	M&S	S-D	Visual	Hom	CL	Dict	Ind
RECONCILE [918]	SL	S-D	Text	Hom	CL	D-B	CI
Agent Laboratory [746]	CTS	S-L	Code, Text	Het	C-O, T-O	Dict	Ind
CoELA[924]	CTS	S-D	Text	Hom	T-O		
The virtual lab [752]	CTS	S-L	Text	Het	C-O, CL	Dict	Ind
SciAgents [743]	CTS	S-L	Text	Het	T-O	Dict	CI
S-Agents [927]	CTS	S-D	Text	Het	T-O, CL	Dict	
GPT-Bargaining [1003]	CTS	S-D	Text	Het	C-O	D-B	CI
FORD [1004]	M&S	S-D	Text	Het	C-O	D-B	CI
MADRA [1005]	CTS	S-D	Text	Het	C-O	D-B	
Multiagent Bench [948]	CTS	S-D	Text	Hom	T-O, CL	D-B	CI, Ind
OASIS [936]	M&S	D	Text	Het	C-O		
S <sup>3</sup> [255]	M&S	S-D	Text	Het	C-O		
FPS [981]	M&S	S-D	Text	Het	C-O		
GPTSwarm [1006]	CTS	D	Code, JSON, Text	Hom	T-O	Dict	CI, Ind
ChatEval [1007]	CTS	D	Text	Hom	T-O	Voting	CI
MetaGPT [626]	CTS	S-L	Code, JSON, Text, Visual	Het	T-O	Dict	CI
AutoAgents [1008]	CTS	D	Text	Het	T-O	C-O	CI
SWE-agent [628]	CTS	D	Text	Hom	T-O	Dict	Ind
AgentCoder [994]	CTS	D	Code, Text	Het	T-O	D-B	CI
MASTER [1009]	CTS	S-L	Text	Hom	T-O	D-B	CI
Reflexion [48]	CTS	D	Text	Het	T-O	D-B	Ind
MACM [1010]	CTS	D	Text, Code	Het	T-O	D-B	CI
Debate [985]	CTS	S-D	Text	Het	C-O	D-B	CI

# Chapter 14

## Communication Topology

### 14.1 System Topologies

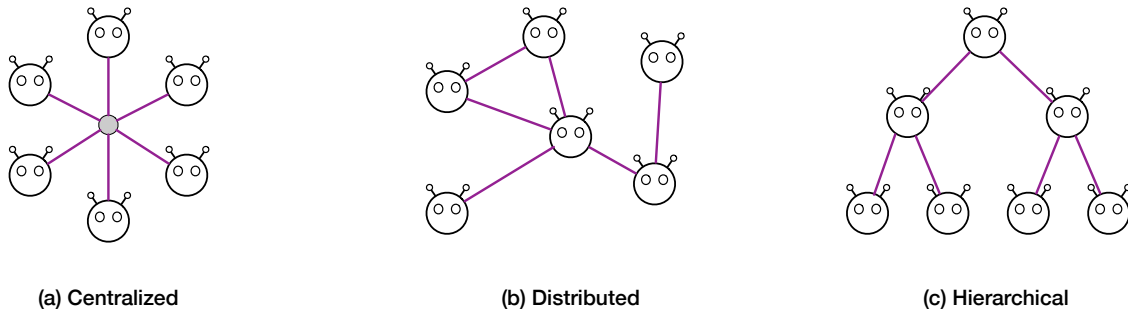


Figure 14.1: Different types of topological structure for multi-agent collaboration.

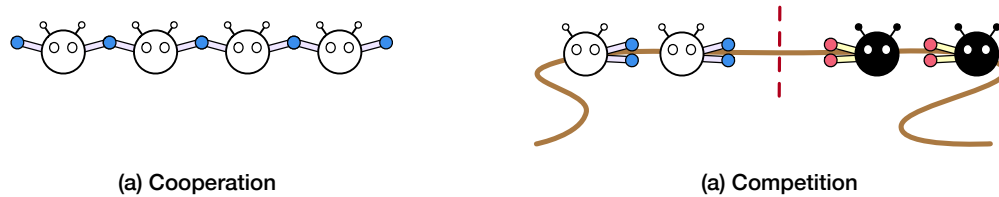


Figure 14.2: Collaborative and competitive agents.

This section examines the interaction typology in LLM-based multi-agent systems (MAS) and its impact on communication, collaboration, and task execution. We first analyze static topologies—where connectivity patterns are fixed by domain knowledge—and then explore dynamic (adaptive) topologies that adjust inter-agent connections based on performance metrics, workload variations, or strategic constraints. We conclude with a discussion of scalability challenges and trade-offs in balancing system cost, performance, and robustness, drawing on recent research in distributed processing, self-organization, and emergent collaborative behaviors.

#### 14.1.1 Static Topologies

Static topologies are defined by predetermined structural patterns that remain largely unchanged during system execution. In these configurations, connections among agents—or between agents and a central coordinator—are established using fixed rules and heuristics, ensuring predictable communication flows and simplified coordination. Three canonical forms are typically considered: layered (hierarchical), decentralized, and centralized architectures.

**Layered (Hierarchical) Structures** Layered topologies arrange agents hierarchically, with high-level agents coordinating or supervising lower-level ones. This approach mirrors traditional management frameworks—such as Standard

Operating Procedures (SOP) or the Waterfall model—where tasks are decomposed into sequential, well-defined stages. For instance, the AutoAgents [1008] framework assigns roles (e.g., Planner, Agent Observer, and Plan Observer) to synthesize execution plans, while ChatDev [983] leverages hierarchical task decomposition to streamline software development [626, 921, 627]. Although hierarchical structures facilitate debugging, performance monitoring, and modularity, they can create bottlenecks when upper-tier agents are overloaded [1011]. Recent studies in storytelling [1012, 1013, 1014] and data science applications including data cleaning [1015, 1016], visualization [1017, 1018] and auto machine learning [1019, 1020], highlight the trade-off between consistency and the emergence of adaptive real-time behaviors.

**Decentralized Structures** In decentralized topologies, agents interact on a peer-to-peer basis without a central coordinator, forming networks that are often modeled as chains, rings, small-world, or random graphs [1021, 971]. This structure enhances fault tolerance since the failure of a single agent does not compromise the network. For example, [1022] show that distributing graph reasoning tasks among multiple agents enables scalability beyond the context length limits of individual LLMs. Additionally, [1023] propose decomposition strategies that allow an orchestrating LLM to delegate subtasks effectively. However, maintaining a coherent global state in decentralized systems necessitates sophisticated consensus and synchronization protocols.

**Centralized Structures** Centralized topologies rely on a master coordinator that gathers information and directs peripheral agents hierarchically. Such a setup allows for better control over handling resources and sharing a global view, such as with culture parks and Lyfe Agents [1024, 1025]. With additional agents, however, a bottleneck at the center node may occur, with increased communication overhead and susceptibility to failures. Current studies on coordinator-agent configurations [971] and research on ensuring autonomy for centralized configurations [1026] point out problems with scalability with consistency. While consistency is guaranteed for centralized architectures, there may not necessarily be flexibility for dynamic adaptation.

Briefly, static topologies have advantages of determinism and predefinition. With pre-defined structural patterns, these systems have predictable communication patterns and effective coordination among agents. Topologies of these structures are typically defined on structural knowledge or static rules, and, as such, they suit domains where workflow for the tasks is static, there are predefined roles, and system requirements are well defined. The second primary advantage is design, implementation, and maintenance ease. With structure predefined, design as well as execution procedures are made simpler, and, as a result, maintenance is a simpler process. Resource handling as well as modularization gets simpler due to well-defined, static structure.

However, static topologies themselves are nonflexible, grounded on pre-specified patterns of connectivity that do not respond to real-time changes. Well suited for a specific purpose at design time but entirely lacking flexibility for reacting to unforeseen challenges, including sudden agent breakdown, varying degrees of task complexity, and system goal modification, static topologies do not have real-time response flexibility potential. Real-time response inflexibility inhibits runtime system reconfiguration and decreases system effectiveness in dynamic settings where circumstances occur. Failure to self-organize and morph according to emerging conditions may equate to inefficiency as well as low system performance, particularly where dynamic or emergent settings are at hand.

#### 14.1.2 Dynamic and Adaptive Topologies

While static topologies provide determinism and predictability—illustrated by static topologies such as hierarchical or centralized ones performing well with stable-task domains and well-defined roles—static topologies do not fit open-ended or novel domains. Real domains, from real-time collaborative plan, to dynamic social simulations, often demand that agents make changes on their patterns of interaction as work continues, available resources vary, or feedback from the environment is received. Such structural tension with adaptive malleability generates dynamic topologies, which, at runtime, recast inter-agent relationships as a response to feedback on performance, workload, or strategic constraints, striking a balance between consistency and responsiveness.

For example, DyLAN framework [725] supports inference-time agent selection through a two-step process: a forward-backward team optimization step with unsupervised Agent Importance Scores, followed by dynamic team reformulation at runtime. Similarly, OPTIMA [1027] optimizes inter-agent connectivity iteratively through a generate-rank-select-train framework, utilizing reward functions as a means for determining a balance among task quality, token efficiency, and readability, with communication actions further optimized through strategies such as Direct Preference optimization. The MAD framework [649] illustrates flexibility through a joint optimization among three prompt phases and structure, with dynamic role assignment (such as verifiers and debate participants) within pruned spaces for structure.

Topological control also becomes tractable through technological advancements. GPTSwarm [651] conceptualizes agents as computation graphs and uses evolutionary strategies and reinforcement learning for adjusting adjacency matrices for optimizing nodes based on feedback for the task. MACNET [1028] uses a directed acyclic graph architecture with supervisory instructors managing edges and executive assistants managing nodes for more complex coordination domains, facilitating adaptive communication through topological ordering and sensitive propagation of output. Application-specific versions also emphasize architecture diversity. Open-world environments have DAMCS [1029], which couples hierarchical knowledge graphs (A-KGMS) with structured communication schemes (S-CS) for co-operative planning as a function of messages passed based on context. AutoAgents [1030] leverages a dynamic drafting-execution pipeline with pre-defined agents jointly sketching out expert teams, a design that’s highly effective for creative applications such as novel generation through parallel processing and internal supervision. Noticeably, small-world development within large-scale MACNET [1028] systems corresponds with graph reasoning ideas shown in [1022], where distributed architecture bypasses local limitations of LLM through structured collaboration. In terms of collaborative task solving, several paradigms have emerged that emphasize the role of dynamic topologies. These paradigms include search-based methodologies, LLM-based generation, and configurations utilizing external parameters.

**Search-based Methods** A number of works adopt search-based methodologies to iteratively optimize communication structures. For example, ADAS [741] employs a Meta Agent Search algorithm that iteratively generates and tests new agent designs within a code space, archiving superior configurations and thereby updating subsequent generation strategies. Similarly, Aflow [773] models each LLM call as a node in a graph and utilizes Monte Carlo Tree Search (MCTS) to dynamically extend and refine the workflow. Other frameworks, such as MAD [1031] and OPTIMA [1027], integrate iterative generate–rank–select–train paradigms that echo MCTS principles to balance task performance with efficiency.

**LLM-based Methods** Complementing search-based methods, several recent works leverage the generative capacity of LLMs to construct and adapt dynamic topologies. Dylan [725] introduces a temporal feed-forward network (T-FFN) model that treats each communication step as a network layer, using forward-backward propagation to compute Agent Importance Scores for dynamic team selection. In related work, DAMCS [1029], AutoAgents [1030], and TDAG [1032] dynamically generate specialized sub-agents or update hierarchical knowledge graphs, enabling cooperative planning and task decomposition. Further, frameworks such as AutoFlow [773] and Flow [1033] represent task workflows in natural language programs or activity vertex graphs (AOV), allowing continuous refinement through reinforcement learning signals. ScoreFlow [788] complements these approaches by applying gradient-based (loss-gradient) optimization to continuously reconfigure agent workflows.

**External Parameters** Given that fine-tuning LLM-based agents is often resource-intensive, a considerable number of researchers advocate configuring inter-agent topologies by training parameters independent of the LLM-agent. This approach is initiated by GPTSwarm [651], in which the inter-agent topologies are represented as a directed acyclic graph (DAG), with edge weights serving as the sole trainable component of the system. Further advancing this paradigm, AgentPrune provides a unified modeling framework from the spatial-temporal graph perspective for mainstream MAS, where communication redundancy, i.e., unnecessary edges, is identified and pruned through magnitude-based pruning. Follow-up works in this line of research include G-Safeguard [1034], which similarly trains GNN outside of the MAS to detect and eliminate malicious communication paths. Although these methods are parameter-efficient, their relatively small parameter space and low coupling with LLM-agents often result in performance limitations to some extent.

**Discussion** Dynamic topologies extend beyond task-solving and play a crucial role in simulating complex social interactions. As detailed in a recent survey [975], LLM-based agent models can evolve inter-agent links to capture real-time changes in autonomy, social behaviors, and environmental feedback across various domains, including cyber, physical, and mixed environments. Systems such as [50], OASIS [936] and ProjectSid [989] simulate dynamic social networks. [50] employs generative natural language memory retrieval to adjust social ties based on agents’ experiences, while OASIS constructs a real-time social media environment with continuously updated user relationships and information flows. Project Sid [989] introduces the PIANO (Parallel Information Aggregation via Neural Orchestration) architecture, enabling over 1,000 autonomous AI agents to interact in real-time within a Minecraft environment, leading to the emergence of complex societal structures such as specialized roles, collective rule adherence, and cultural and religious transmission. Additionally, architectures like AgentScope-scability [1035] and Social Survey [975] support large-scale multi-agent simulations, enabling studies of cultural dissemination, collective decision-making, and emergent group dynamics in environments with hundreds or thousands of interacting agents. Additionally, dynamic topologies are also tailored to specific application domains such as medical and open-domain embodied AI. In the medical field, AI hospital [1036] and agent hospital [921] simulate real medical workflows, where iterative cycles of diagnosis, treatment, and feedback continuously reshape communication patterns among various roles, such as intern doctors,



patients, examiners, and supervising physicians. These frameworks dynamically adjust inter-agent communication to optimize collaboration and decision-making. Similarly, in open-domain and embodied AI applications, frameworks like IOA [933] support heterogeneous, cross-device agent interactions, facilitating dynamic team formation and task allocation in real-world scenarios.

Although the aforementioned dynamic multi-agent topologies have made substantial progress in performance metrics, they still face the following three limitations, which we believe should be the focal points for future research on dynamic topologies:

**(1) Generalizability.** Current MAS topologies are typically optimized for a single-task domain. For example, AFlow [773] is dedicated to search and optimization within math or code benchmarks, producing a fixed workflow that is difficult to adapt to new task domains. Other dynamic topologies, such as ADAS [741], GPTSWarm [651], and AgentPrune, face the same challenge. We argue that MAS should be capable of lifelong learning, wherein the system generalizes across different task domains with minimal resources (e.g., API calls, FLOPs, GPU hours).

**(2) Resource Efficiency.** Present dynamic topologies often tend to optimize for complex, resource-intensive structures. Their training processes are typically exorbitantly costly, as exemplified by ADAS [741], where training with GPT-3.5 incurs a cost of approximately \$300 per session. Such expenses severely constrain their large-scale applicability in real-world scenarios. Future developments should focus on achieving better test-time topology optimization with significantly reduced costs.

**(3) Inference Efficiency.** As MaAS [787] has incisively observed, multi-agent topologies of excessive complexity, while capable of consistently delivering satisfactory performance, are lamentably deficient in *task adaptability*. That is to say, they are unable to dynamically allocate reasoning resources (i.e., tools, the number of agents, and reasoning steps) in response to the difficulty of a given task. Consequently, this may lead to a certain lack of efficiency in the inference process. Although MaAS has, to a certain extent, achieved task dynamism through the designed agentic supernet, their applicability and scalability in large-scale deployment still remain to be tested.

## 14.2 Scalability Considerations

Scalability is a critical challenge in LLM-based multi-agent systems (MAS), especially as the number of agents grows. In fully connected networks, the number of communication paths grows quadratically, leading to a communication explosion that increases token usage and computational costs [1037, 626]. Centralized and layered topologies can experience synchronization bottlenecks if supervisory nodes are inundated by messages, whereas decentralized networks—while more fault tolerant—necessitate complex consensus algorithms to achieve a coherent global state.

Recent work such as [1028] demonstrates that when multi-agent collaboration is structured as a directed acyclic graph (DAG), the system can scale efficiently to handle large graphs—up to 1,000 nodes or more—without significant performance degradation. Similarly [1022] shows that distributing graph reasoning tasks among many agents circumvents the limitations imposed by long textual inputs and context-length constraints. Moreover, studies on self-organized agents [1038] reveal that dynamic multiplication and task distribution allow the system to maintain a constant workload per agent while increasing overall processing capacity. Finally, the multi-dimensional taxonomy proposed by [1039] provides a valuable framework for analyzing trade-offs between agent autonomy and alignment, offering insights into how to balance centralized control with decentralized flexibility to optimize scalability.

In addition to these foundational studies, recent advances in practical multi-agent platform design further enrich the scalability discussion. For example, AgentScope [1035] offers a developer-centric platform that leverages an actor-based distributed framework to enable seamless migration between local and distributed deployments. Its unified workflow and automatic parallel optimization significantly reduce the communication overhead and synchronization challenges that typically emerge as agent numbers increase. By incorporating fault-tolerance mechanisms and intelligent message filtering, AgentScope illustrates how system-level supports can be designed to maintain performance even in dynamic and heterogeneous deployment environments.

Another complementary approach is presented in Project Sid [989], which explores scalability within the realm of simulating agent civilizations. Here, the focus shifts from isolated task solving to the simulation of complex societal dynamics. The proposed PIANO (Parallel Information Aggregation via Neural Orchestration) architecture allows agents to operate concurrently by decoupling slower cognitive processes from rapid reactive modules. A dedicated cognitive controller is introduced to ensure coherence among multiple parallel outputs. This design not only enables scalability from small groups to simulations involving over a thousand agents but also effectively addresses the inherent coordination challenges arising from high-frequency interactions.



Taking scalability to an even larger scale, AgentSociety [1040] demonstrates a comprehensive framework for simulating realistic social environments with up to 10,000 agents. By integrating LLM-driven social generative agents within a realistic urban, social, and economic setting, AgentSociety employs distributed computing and a high-performance messaging system (e.g., MQTT) to support millions of daily interactions. This platform exemplifies how emerging hybrid architectures can support macro-level phenomena—such as economic market dynamics, opinion diffusion, and urban planning simulations—by effectively managing the trade-offs between communication cost, coordination overhead, and emergent behavior fidelity.

Despite the theoretical advantages of scaling up agent populations, it is imperative to question whether pursuit of large-scale agent deployments is inherently valuable for all task-solving scenarios. Although the total computational capacity scales with the number of agents, when memory overhead and inter-agent communication costs are factored in, the marginal utility of adding additional agents may demonstrate diminishing returns. This phenomenon arises from the fundamental constraint that, while the overall workload is the product of individual task complexity and the degree of labor division, coordination costs tend to increase super-linearly with agent count. Therefore, for many bounded problem domains, there is likely an optimal agent population size beyond which performance plateaus—or even deteriorates—due to excessive coordination overhead.

Conversely, in simulation scenarios where the objective is to model complex social dynamics, emergent behaviors, or large-scale collective intelligence, scaling to numerous agents becomes not merely beneficial but essential. In these contexts, the research focus shifts from optimizing computational efficiency for task solving to accurately reproducing or predicting macro-level patterns emerging from micro-level agent interactions. Such simulations—covering domains like economic market behavior, social network evolution, and urban infrastructure planning—often require the computational overhead of managing vast agent populations in order to capture realistic population-level phenomena.

Hybrid architectures that combine centralized oversight with decentralized sub-teams offer a promising solution to these scalability challenges [921, 918]. In these designs, supervisory agents handle global objectives and coordination, while worker agents focus on executing specific subtasks. This hierarchical organization helps to mitigate information overload at any single node and allows for dynamic adjustment of agent team sizes based on task demands, thereby optimizing resource utilization. Furthermore, advanced techniques such as graph search algorithms, reinforcement learning-based updates, and evolutionary methods are critical for iteratively refining the network structure as the system scales. Intelligent message filtering, prioritization, and aggregation mechanisms can significantly reduce communication overhead without sacrificing the quality of inter-agent collaboration. In addition, asynchronous communication protocols and partial knowledge sharing strategies show promise in minimizing coordination bottlenecks while maintaining sufficient global awareness among agents.

**Concluding Remarks on Scalability** Overall, the study of system topology and scalability in LLM-based MAS reveals a spectrum of design choices—from static configurations that offer simplicity and predictability to dynamic architectures that provide flexibility and adaptability. While foundational works (e.g., [1028], [1038]) emphasize scalable graph structures and self-organizing principles, the practical advances demonstrated by AgentScope, Project Sid, and AgentSociety illustrate how integrated distributed frameworks, concurrent processing, and realistic environment simulations can collectively address the challenges of scaling multi-agent systems. The context-dependent nature of scalability requirements—contrasting between task-solving and simulation scenarios—highlights the importance of purpose-specific design in multi-agent architectures. As research continues to evolve, the development of more sophisticated adaptive algorithms, distributed architectures, and multi-dimensional evaluation frameworks will be essential for advancing the scalability and practical viability of LLM-based multi-agent systems.

## Chapter 15

# Collaboration Paradigms and Collaborative Mechanisms

In this chapter, we offer a detailed exploration of these purposeful interactions, examining how one agent influences collaboration within MAS. We reference the diverse interaction behaviors that emerge from human social structures, further explaining multi-agent collaboration through interaction purposes, interaction forms, and the relationships that form.

Multi-Agent Systems (MAS) comprise multiple agents that interact in a shared environment, autonomously making decisions to accomplish tasks collaboratively or compete with each other [1041]. In our context, we focus on collaborative phenomena because they widely appeared in most practical applications. Basically, each agent in MAS is equipped with different roles and initial knowledge and its own set of goals.

When engaged in problem solving or communication, agents interact with other agents or the environment to collect and process information, independently making decisions based on their objectives, existing knowledge, and observations, and subsequently executing actions [975, 1041, 1042, 1043]. Knowledge, memory, and environmental observations form the agents' beliefs, while varying motivations influence their approach to tasks and decision making [1041]. Consequently, effective problem solving requires diverse purposeful interactions, including agent-agent and agent-environment. These interactions may involve multiple rounds and occur in various directions, depending on the system design.

### 15.1 Agent-Agent collaboration

Considering the categorizations of MAS collaborations, we focus on more details on the granularity needed to capture the nuanced dynamics in complex multi-agent interactions. Specifically, we categorize inter-agent interactions into four types, inspired by sociological insights from human-to-human interaction patterns and applying them to agent-agent interactions in MAS. Sociological theories on human interaction, which include **consensus building**, **skill learning**, **teaching**, and **task division collaboration**, provide a more refined way of classifying agents' interactions. These interactions form collaborative paradigms, which enable diverse intelligent agents to work together effectively in solving complex problems, and they are shaped by various forms of goals, contexts and outcomes. Each paradigm addresses unique challenges related to cooperation, competition, coordination, and decision-making. Additionally, MAS implementations involve agents with different types of interactions, rather than a single type or unidirectional process, forming complex interaction networks that evolve over time. In collaborative software development [626, 627], a senior developer agent may interact task-wise with an architect agent, guide junior agents through multi-round dialogues. They work together on code reviews for decision-making and learn with a testing expert agent to improve test coverage. Examining the objectives and results of these interactions reveals the crucial techniques and technologies shaping agent behavior and decision-making, thereby enhancing our comprehension of multi-agent dynamics.

**Consensus-oriented Interaction** Consensus-oriented interactions concentrate on harmonizing the MAS's final target via negotiation, voting, and social choice frameworks [1044]. This interaction is significant for incorporating diverse knowledge and ensuring agents shift their views towards a unified understanding to achieve consensus [1045]. In this interaction, agents integrate knowledge to establish a unified understanding, which largely helps joint decision-

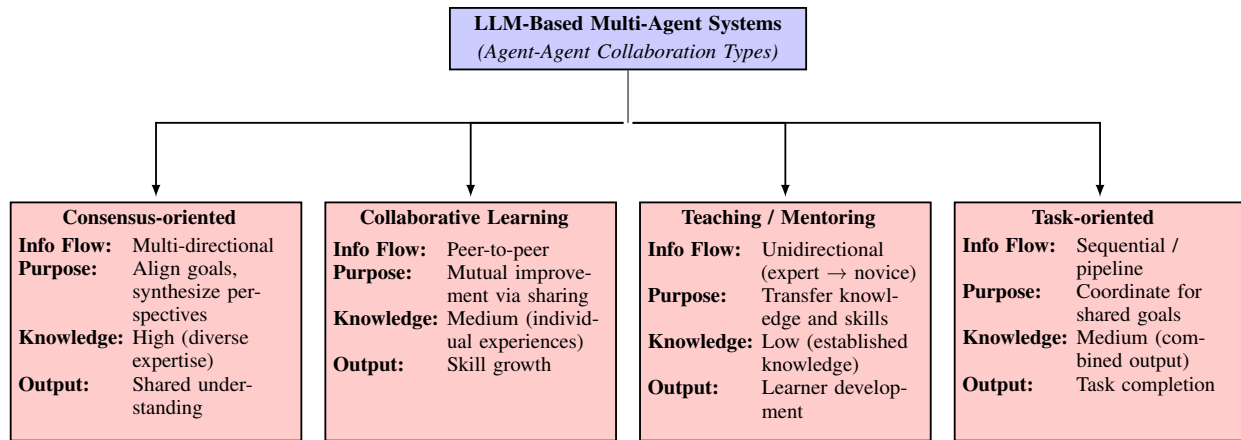


Figure 15.1: An overview of four agent-agent collaboration types in LLM-based MAS: *Consensus-oriented*, *Collaborative Learning*, *Teaching/Mentoring*, and *Task-oriented*. Each type is described along four key dimensions: information flow, collaboration purpose, knowledge integration, and output focus.

making in complex problem-solving situations that demand different viewpoints. For instance, MedAgents [922], MDAgents [1046], and AI Hospital [1036] demonstrate how collaborative dialogue among multidisciplinary agents improves problem solving by sharpening reasoning skills and accessing inherent knowledge.

These dialogues allow agents to ensemble expertise into coherent outcomes, frequently outperforming conventional methods like zero-shot or few-shot reasoning. The importance of consensus-driven teamwork is particularly evident in scientific environments, where addressing complex challenges requires diverse perspectives and meticulous validation. Agent Laboratory [746], serves as an example where PhD and postdoctoral agents collaborate to agree on research objectives, interpret experiments, and consolidate research findings. Similarly, Virutal Lab [752] organize a series of team to conducts scientific research, where all agents discuss a scientific agenda, and individual meetings, where an agent accomplishes a specific task.

Methods for multi-agent consensus typically include several approaches, including **Discussing**, **debating**, **negotiating**, **reflecting**, and **voting**. Common methods for reaching consensus encompass an array of structured techniques. The primary mechanisms involved are **discussing**, **debating**, **negotiating**, **reflecting**, and **voting**. Debates allow agents to obtain competing hypotheses, while negotiation helps resolve conflicting priorities and resource limitations. Specific frameworks have been created to support these consensus-building activities. During these processes, agents gather outputs from peers tackling the same issue, and include environmental feedback as numerical data and contextual details. These interactions enable agents to share viewpoints, assumptions, and progressively achieve a common understanding.

For example, GPTSwarm [651] formulates the collaboration between agents with graph design, that the information flow and edge connections build the basic group discussion. In GPTSwarm, if an agent consistently provides incorrect opinions, it will be excluded. RECONCILE [918] uses a round-table discussion format with several discussion cycles and voting systems based on confidence levels. It integrates reflection by learning from past discussions, using confidence metrics and human insights to improve their responses. Furthermore, debates are quite important for achieving agreement, reducing hallucinations and also addressing complex issues [985, 1047, 1031, 1003]. In GOVSIM [1048], agents collaborate to achieve a balance, and it suggests using a shared resource and conserving it for future needs. The negotiations went beyond simple information exchange and relationship-focused interactions. The Multi-Agent Debate (MAD) framework [1031] promotes creative thinking by having agents deliver arguments in a “tit-for-tat” pattern, with a judge overseeing the process to finalize a solution. The Formal Debate framework (FORD) [1004] enhances consistency among language models through organized debates, enabling stronger models to steer consensus, while weaker ones adjust their perspectives. Similarly, AutoAgents [1030] define a collaborative refinement action in which each agent updates its chat record. In the process, it also appends the previous statements of the other agent and refines its action to achieve consensus.

**Collaborative Learning Interaction** In collaborative learning, interaction usually happens among similar agents. Although architecturally alike, accumulate distinct memories and experiences due to their unique behaviors and varied environmental interactions. By solving problems together, these agents share experiences to boost their strategy learning, task-solving, and skill acquisition capabilities. Over time, each agent enhances its skills through ongoing interaction, leading to the evolution of individuals. The key difference between collaborative learning and consensus-

oriented interactions lies in their fundamental goals and processes. While consensus-oriented interaction focuses on knowledge integration and belief alignment through synthesizing diverse viewpoints to reach agreement, collaborative learning interaction emphasizes peer knowledge construction and experience sharing, prioritizing mutual improvement and individual growth. When engaged in collaborative learning interaction, agents update their context or memory from observing others' behavior. For example, agents can learn optimal strategies by observing the delivery from peers, adapting their own approach based on these observations without necessarily agreeing on a single "best" strategy [961, 962, 963, 971, 965, 967, 972, 968, 969]. As highlighted in [966], the effective discussion tactics significantly impact learning outcomes among agents. In these interactions, agents collaborate to learn and address problems, focusing on mutual understanding and enhancement rather than reaching unanimous decisions. This method refines personal responses and knowledge via ongoing feedback.

The methods commonly employed in collaborative learning interaction include: **1). Experience sharing.** Agents exchange personal insights and best practices. As described in [303], iterative experience refinement enables LLM agents to achieve adaptive improvement in software development via continual acquisition and utilization of team experience in successive pattern and the cumulative pattern. Furthermore, MAS-CTC [301] is a scalable multi-team framework that enables orchestrated teams to jointly propose various decisions and communicate with their insights in a cross-team collaboration environment. It enables different teams to concurrently propose various task-oriented decisions as insights, and then communicate for insights interchange in important phases (multi-team aggregation). Different agent teams utilize a greedy pruning mechanism and aggregation mechanisms to eliminate low-quality content, thus improve the performance in software development. Differently, in MOBA [1049], a novel MLLM-based mobile multi-agent system, global agent reflects on local agent execution results to support adaptive planning to align with the environment. AutoAgents [1030] employs a knowledge sharing mechanism where agents exchange execution results to enhance communication and feedback, where agents can obtain long-term, short-term and dynamic memory from others. **2). Peer discussions.** Peer discussions allow agents to articulate their reasoning processes and learn from others' approaches. MEDCO [923] create a dynamic environment where clinical reasoning and decision-making skills are strengthened through collaborative problem-solving among student agents. Moreover, In [1050], agents engage in structured peer discussions after initializing their output, reviewing each other's reasoning step by step. Through feedback exchange and confidence scoring, agents refine their decision-making, learn from diverse approaches, and iteratively enhance their reasoning accuracy, fostering collaborative knowledge acquisition. **3). Observational learning.** Observational learning occurs when agents monitor others' behaviors and outcomes to inform their own strategies. AgentCourt [1051] develops lawyer agents that participate in court debates and improve through accumulated experiences, demonstrating improved reasoning and consistency through experiential learning. In iAgents [1046], the human social network is mirrored in the agent network, where agents proactively exchange human information necessary for task resolution, thereby overcoming information asymmetry. iAgents employs a novel agent reasoning mechanism, InfoNav, to navigate agents' communication towards effective information exchange. Together with InfoNav, iAgents organizes human information in a mixed memory to provide agents with accurate and comprehensive information for exchange. Additional experimental phenomenon indicates difficulty of certain tasks making agents continuously refine their strategies in pursuit of the required information. MARBLE [948] designs a cognitive evolve planning combining the 'expectation' of the agent and its actual action results to update the overall planning experience for better planning in the next round.

Despite its benefits, collaborative learning interaction faces several challenges. These include ensuring equitable knowledge exchange among agents with varying capabilities, preventing the propagation of errors or biases across the system, maintaining agent diversity while facilitating learning, and developing effective mechanisms for agents to selectively incorporate others' knowledge based on relevance and reliability. Overcoming these challenges requires the meticulous creation of interaction frameworks and learning strategies. And it should balance individual advancement with the broader development of the system. Although issues such as knowledge fairness, bias propagation, and scalability present difficulties, there is great potential to improve MAS, particularly in dynamic and complex environments. By using iterative learning processes and providing opportunities, collaborative learning enables agents to develop richer knowledge bases and more refined problem-solving abilities.

**Teaching/Mentoring Interaction** To tackle these challenges, it is important to carefully develop interaction protocols and learning frameworks that harmonize individual development with overall system progress. In the context of MAS, teaching and mentoring interactions are fundamental mechanisms in collaborative environments, especially in scenarios where knowledge transfer is essential for growth and collective intelligence. Unlike collaborative learning, where knowledge is exchanged reciprocally among agents, teaching and mentoring interactions focus on the unidirectional flow of knowledge from an experienced agent to a less experienced one. The mechanisms and methods used in teaching/mentoring interactions include several key strategies:

- **Criticism and Feedback.** The mentor agent evaluates the learner’s performance and provides corrective or constructive feedback. This helps the learner refine their knowledge and skills through a feedback loop where they update their internal knowledge based on the feedback received.
- **Evaluation.** Mentors assess the learner’s capabilities or progress through performance reviews and clear assessment criteria, providing valuable insights for development.
- **Instruction and Teaching.** Mentors convey targeted knowledge, guidelines, or techniques using direct instruction which allow learners to pose questions and receive clarifications.

**Iterative Teaching and Reinforcement** Teaching is typically progressive, where each phase provides opportunities for the learner to complete tasks and get feedback. For example, in the MEDCO system [923], student agents improve their professional skills through a cyclic practice-oriented learning approach directed by expert mentors, in addition to engaging in peer discussions. These expert agents conduct ongoing assessments and provide real-time guidance on clinical competencies, focusing on patient interaction skills and diagnostic reasoning. [921] shows that an agentic doctor can continually improve their diagnosis by merely interacting with agentic patients in a simulated hospital and can transfer its learned knowledge of real-world cases.

This interaction type can be categorized based on the direction of knowledge transfer into two primary types: unidirectional and interactive. Unidirectional is rooted in traditional teaching models where knowledge flows from the teacher to the student. This approach emphasizes the transmission of facts and concepts, often involving lectures and direct instructions [923].

**Task-oriented Interaction.** Task-oriented collaborations involve agents working together to achieve common objectives through effective coordination and task decomposition strategies, as well as a high degree of cooperation and coordination. Agents interact primarily by processing upstream output and generating results for downstream agents following established task dependencies rather than engaging in complex discussions or debates.

Recent frameworks demonstrate diverse implementations of this interaction pattern: **(1) software development frameworks** such as MetaGPT [626] and ChatDev [627], agents operate in a structured pipeline that mirrors the software development lifecycle. For example, architect agents process requirements to generate technical specifications, which development agents then use to produce code, followed by testing agents who validate the implementations; **(2) Collaborative reasoning** frameworks like Exchange-of-Thought (EoT) [1052], GPTSwarm [651], MACNET [1028] involve structuring agents in a specific format (e.g., ring, tree, directed acyclic graphs, optimizable graphs), which mitigates context expansion risks by ensuring only optimized solutions progress through the sequence, and enforcing multiple agents to collaborate together towards solving complex mathematical or knowledge reasoning tasks; In **(3) ML applications** [1053, 1019], agents adhere to stringent workflow structures, each fulfilling specific tasks in processes. For more complex tasks such as VideoQA, the TravelER framework [1054] showcases modular task breakdown across structured phases (Traverse, Locate, Evaluate, and Replan), with a Planner agent managing interactions and improving strategies based on iterative agent inputs.

These handoffs rely on explicit deliverables instead of direct agent negotiations. Inspired by GPTSwarm [651]-like graph agentic systems, MACNET [1028] structures agents into directed acyclic graphs (DAG). Here, supervisory figures issue directives while executors implement solutions. By ensuring only optimized solutions progress through the sequence, this setup mitigates context expansion risks. In ML applications [1053, 1019], agents adhere to stringent workflow structures, each fulfilling specific tasks in processes. For more complex tasks such as VideoQA, the TravelER framework [1054] showcases modular task breakdown across structured phases (Traverse, Locate, Evaluate, and Replan), with a Planner agent managing interactions and improving strategies based on iterative agent inputs.

Beyond organized development, task-driven interactions have been shown in open-ended contexts such as Minecraft game, in where agents adjust to ever-changing environments. In [927], leader agents manage workflows by breaking down complex objectives into specific tasks, while executor agents perform actions like gathering resources. Coordination mechanisms are important for ensuring agents collaborate effectively towards final goal, including communication protocols, synchronization strategies, and resource-sharing techniques. The interaction of agents in MAS for task execution has garnered significant interest, notably through utilizing LLMs for handling intricate tasks and workflows. The collaboration of agents are vital for task completion, particularly in ever-changing settings like software development and project management [626, 630].

## 15.2 Human-AI Collaboration

To unlock the potential of MAS in meeting human objectives, people often work alongside them using three primary methods: **one-off task delegation**, **multi-turn interactive instruction**, and **immersive human-agent collaboration**.



In **one-off task delegation**, humans delegate single-instance tasks to MAS, such as posing a question to a Q&A platform or assigning a coding task [1055, 626]. Without additional input, the agent handles the task autonomously, delivering a complete response or solution in a single reply. This is presently the prevalent way humans collaborate with LLM-based agents [922, 627, 31].

For **multi-turn interactive instruction**, humans engage in iterative interactions with LLM-based agent systems to refine and explore solutions until a satisfactory result is achieved. This type of interaction is widely seen in creative applications, such as image editing or writing edit [938]. For instance, a user might ask the system to add an object to a specific location in an image, replace an element, change the background, or revise a part in a sentence. These interactions often span multiple rounds, with users continuously refining their requests until the desired outcome is reached. Moreover, certain other LLM-based agent systems may require human approval or clarification during multi-turn interactions before proceeding to the next step [1056, 930]. Under human guidance, these LLM-based agent systems can complete household tasks as well as software development tasks.

**Immersive human-agent collaboration** features LLM-based agents simulating human behaviors to serve as partners. For instance, in an immersive setting, humans treat these agents as teammates, achieving common objectives. Instances include agents representing humans in meetings or help solve tasks like chores or projects. This strategy highlights effective integration and teamwork in dynamic contexts [937, 924].

To assess Human-AI collaboration quantitatively, several frameworks have been suggested. Co-Gym [1057], for instance, measures the communication, situational awareness, and personalization of LLM-based agents in tasks such as travel planning, writing related work, and tabular analysis.

In summary, as LLM-based agent systems have advanced, Human-AI collaboration has diversified to address challenges across domains. This ranges from simple command-based AI interactions for questions, to multi-turn dialogues for design and development, and partnering with human daily tasks.

With advancements in LLM-based agent systems, they are expected to integrate more into daily life, streamlining tasks and boosting efficiency. At the same time, humans will refine and adapt their ways of interacting with AI, leading to more effective collaboration. We believe this shift will drive fundamental changes in both social productivity and the social relations of production, reshaping how work is organized and how humans and AI cooperate in the large language models era.

### 15.3 Collaborative Decision-Making

Collaborative decision-making processes are crucial for ensuring the efficient operation of MAS and the successful completion of tasks. Although collaboration itself is a core feature, the approaches of decision-making directly determines the effectiveness of collaboration and the overall performance of the system. Recent research has highlighted the critical role of collaborative decision-making. [1037] showed that diverse decision-making methods can significantly enhance the collaborative efficiency of the system. [649] emphasized that a rational decision-making mechanism can stimulate the emergence of intelligence within a system.

From a broader perspective, the collaborative decision-making process can be divided into two major categories based on their architectural characteristics: Dictatorial Decision-Making and Collective Decision-Making [1037].

**Dictatorial Decision-Making.** Dictatorial Decision-Making is a process where decision-making relies on a single agent in a MAS. In this paradigm, all agents send their state information or local observations to this dictatorial agent. The dictatorial agent is responsible for assembling this data, studying the core problems, and establishing definitive decision guidelines. The key principle for such an approach is to leverage a global mindset in moving towards improved decision-making, hence paving the reliability of the system performance along with the successful achievement of task goals. [1031, 1058, 1046] demonstrated the single-agent decision-making process with a single LLM, who synthesized various views on the same problem to make decision-making even more objective and comprehensive. Furthermore, [134, 1059] suggested the weighted integration method through ranking, scoring or checklist, enhancing the robustness of decision-making procedures. In addition, beyond the explicit inclusion of perspectives, [1030, 1060] proposed architectures where a central agent breaks down complex tasks into simpler sub-tasks and assigns them to specialized agents grouped by their functionalities. Moreover, in [651, 1028], it is common that the last node's agent works in an environment to assemble the past information and deduce a conclusion according to the topological structure, rather than by a central agent.

**Collective Decision-Making.** Collective Decision-Making involves agents collaborating to reach decisions without a central authority, relying on local data and interactions like voting or negotiation. This method shares decision-making power among agents, allowing the system to adapt according to changes while maintaining robustness and scalability.

- ***Voting-based Decision Making*** Voting systems are important for collective decision-making, providing a framework for reaching consensus. A conclusive majority is achieved through voting as described by [1045, 968]. Moreover, the GEDI electoral module [1037] enables multiple voting methods. This method largely improve reasoning and fault-tolerance while avoiding complex system designs.
- ***Debate-based Decision Making*** In comparison with voting-based methods, debate-based decision-making focuses on organized interactions between agents, in order to obtain the best result. In [1031, 1061], agents participate in guided discussion, where they articulate and proposals in an attempt to resolve disagreements and reconcile points of view. Simultaneously, [1050, 1062] practice restraint stance, using communication channels among agents for consensus-building through repeated discussions. To tackle the issue of “cognitive islands,” certain systems would employ a common retrieval knowledge base to enable agents to be aware of the same knowledge throughout debates [1005]. By mimicking human dialogue, these systems allowed agents to exchange perspectives and make more informed decisions.

**Discussion and Future Work** Collaboration in multi-agent systems (MAS) still faces numerous challenges that require further research. Current methods are largely based on contextually dependent interactions; however, they do not include a specific framework for training and optimizing cooperative actions. This heavy dependence on large language models (LLMs) has some limitations, as their effectiveness is inherently tied to the size of the LLM’s contextual window and its native reasoning capabilities. While LLMs provide a solid foundation for enabling interactions, these systems are still limited by the inherent limitations of context-dependent communication.

Future studies should focus on finding frameworks that inspire agents for active learning with regard to optimal timing and information dissemination methodologies. Using methodologies from multi-agent reinforcement learning (MARL), there is a growing requirement for strategies that will help agents determine appropriate moments for information sharing, as well as what information should be shared through what channels. This calls for not just devising novel interaction protocols but also incorporating training methodologies that will constantly optimize these protocols with each improvement.



## Chapter 16

# Collective Intelligence and Adaptation

The concept of collective intelligence is central to the development of multi-agent systems(MAS), drawing inspiration from biological and societal cooperation. An inherent concept within collective intelligence is the “Wisdom of Crowds” by [915], which asserts that independent communities often make better decisions as a whole than any one person. Cognitive theoretical models like the Society of Mind [17] and its related theory mind [916, 917] further support the paradigm, suggesting that intelligence springs from a synergy among primary, specialist components. Moreover, In human societies, individuals collaborate, divide labor, and engage in collective problem-solving to address complex challenges. MAS adopt similar strategies where specialized agents to participate in solving complex problems and collective decision-making [914].

The emergence of collective intelligence within MAS is a dynamic and iterative process. Through continuous interaction, agents develop a shared understanding and collective memory progressively. The interaction dynamics are strengthened by heterogeneity among individual agents, environmental feedback, and agent-agent interactions [914], which are all important for the emergence of complex social networks and improving decision-making strategies. It is worth highlighting that collective intelligence is not merely the summation of individual capability, but refers to emergent behavior beyond individual agent capacity. beyond individual agent capacity. Individual agent development is deeply linked with collective intelligence growth. With ongoing involvement with collective tasks, and self-reflection on shared contexts, agents increasingly develop reasoning and decision-making capabilities. The evolution of individual agents is closely related to collective intelligence evolution. Through continuous interaction in joint activities and critical examination of shared contexts, agents continuously refine their reasoning and decision-making abilities.

In parallel, complex and diverse behavior among agents emerges. These include beyond-restricted-protocol behaviors, such as advanced social interactions, including trust, strategic deception, adaptive camouflage, and emergent cooperation, evoking a shift from reactive into cooperative strategies, as well as deeper social dynamics. With a chain of recursive interactions, agents necessarily form cooperative strategies, which eventually turn into social contracts, organizational hierarchies, and divisions of labor. Social phenomena necessarily emerge through recursive interactions among agents, coupled with their adjustment with the changing environment. It marks a transition from fundamental cooperative behavior into complex social constructs, leading to cultural norms and conventions.

### 16.1 Collective Intelligence

The concept of collective intelligence, which refers to the ability of a group of agents to exhibit problem-solving capabilities that surpass those of individual agents. This phenomenon is often characterized by emergent behaviors, sophisticated decision-making, and higher-order reasoning abilities that arise from interactions among agents, leading to enhanced performance in collaborative decision-making scenarios and social simulations [975]. [917] demonstrate that LLM-based agents can exhibit collaborative behaviors and high-order Theory of Mind capabilities, which are crucial for understanding the perspectives of other agents in a shared environment. Their findings suggest that the integration of LLMs into MAS can facilitate more sophisticated forms of collective intelligence, thereby improving the overall efficacy of collaborative decision-making.

**Improved System Performance** A primary advantage of collective intelligence in MAS is that collaboration leads to superior problem-solving capabilities. Collective intelligence can be encouraged to overcome “groupthink” and individual cognitive bias in order to allow a collective to cooperate on one process – while achieving enhanced

intellectual performance. When individual agents share information and coordinate actions, the system can achieve better results than any single agent operating independently [626, 922, 1046, 1031, 1063]. Collective intelligence is therefore shared or group intelligence that emerges from the collaboration, collective efforts, and competition of many individuals and appears in consensus decision making. Collective intelligence strongly contributes to the shift of knowledge and power from the individual to the collective. [924] demonstrated this through their Cooperative Embodied Language Agent (CoELA), which achieved a 40% improvement in efficiency over traditional planning methods in ThreeDWorld multi-agent transport tasks. This substantial improvement stems from the system’s ability to effectively utilize LLMs for planning and communication in multi-agent settings, providing compelling evidence for enhanced collaborative decision-making capabilities. As previously discussed, the inherent diversity and interdisciplinary nature of LLM-based multi-agent systems, along with various inter-agent interaction, which provide internal feedback and enriched context for individual decision-making, hence reduce bias and improve the consistency of solution [918].

**Emergent Behaviors** One of the most intriguing aspects of collective intelligence is the emergence of new, complex behaviors that arise spontaneously from agent interactions. These behaviors are not explicitly programmed but emerge from learning and adaptation. As discussed in various studies [971, 965, 966], agents developed strategic behaviors, including trust-building, adversarial tactics, deception, and leadership during the game. The collective behavior evolved through experience sharing, where village-aligned agents learned cooperation and strategic alliance formation, and wolf-aligned agents improved deception through “information confusion” tactics. Moreover, agents optimized voting patterns and deception strategies without explicit training, which indicates the group intelligence emerged over multiple rounds of interactions. Similarly, in the Avalon game [968], researchers observed that agents became better at identifying and countering deceptive information. Individuals adapted to deceptive environments and refined their decision-making using first- and second-order perspective shifts. Furthermore, agents demonstrated adaptive cooperation and ad hoc teamwork, despite no predefined collaboration protocols [969]. These findings highlight the ability of LLM-based agents to develop sophisticated behaviors through interaction and learning, showcasing the potential for emergent behaviors in collective intelligence scenarios. Notably, these emergent behaviors rely on memory and reflective mechanisms. Agents retrieve and reflect on historical information to generate a compact context, enhancing their reasoning capabilities [239]. In MAS, shared context and environmental information significantly boost agents’ usable memory. This enables agents to build on past interactions, refine strategies, and adapt more effectively to dynamic environments [1064].

**Social Evolution** One of the most significant findings in the field of generative agent societies is the spontaneous emergence of social norms. [1065] demonstrated that agents, through continuous interaction, are capable of creating, representing, spreading, evaluating, and complying with social norms. These norms serve as the foundation for social order, reducing conflicts and improving coordination among agents, thereby leading to more stable and organized societies. Interestingly, the study found that agents develop norms more rapidly in their beliefs than they do in their behaviors. This suggests that while agents may quickly internalize certain norms, the translation of these norms into consistent actions takes longer. Over time, these norms tend to synthesize into more general principles, resulting in more concise and effective personal norm sets. Furthermore, the Project Sid simulation[989] models large-scale agent societies and provides further evidence of the emergence of social norms and role specialization. In this study, agents were observed to autonomously form specialized social roles. These roles were not predefined but emerged naturally as agents interacted within their environment and developed collective rules. The simulation also highlighted the importance of democratic processes in the adherence and modification of these collective rules. Agents were found to engage in cultural and religious transmission, spreading ideas and doctrines across communities. This process of norm creation and role specialization leads to better organization, reduced conflict, and adaptive governance structures within the society. The evolution of cultural and religious beliefs in multi-agent societies is also observed in [1066], which occurs through agent-driven selection of ideas, mirroring real-world societal changes. Additionally, the [936], which simulates social interactions among one million agents, provides valuable insights into cultural transmission and group polarization. Cultural memes and belief systems propagate naturally among agent societies. Agents exhibit herd behavior, conforming to prevailing opinions even when these opinions are irrational. This leads to the emergence of group polarization, where agents reinforce extreme views through repeated interactions. This finding highlights the significant impact of group size on the dynamics of cultural evolution and social behavior.

## 16.2 Individual Adaptability

In multi-agent systems (MAS), individual adaptability refers to an agent’s ability to adjust its behavior and decision-making strategies based on previous interactions and experiences. This is also defined as self-evolving, where agents can dynamically self-evolve by modifying themselves, such as altering their initial goals and planning strategies, and training themselves based on feedback or communication logs [38]. This adaptability is facilitated by the integration of large language models (LLMs), which support dynamic monitoring and adaptation processes [1067], as well as the

agents' memory capabilities and information exchange. These modules are crucial to ensure that agents can continuously improve their performance, respond effectively to dynamic environments, and optimize their decision-making processes. We categorize the mechanisms contributing to individual adaptability into memory-based learning and parameter-based learning, where there are training-free and training-based approaches.

**Memory-based learning** Memory and reflective mechanisms significantly enhance individual adaptability in LLM-based multi-agent systems by leveraging historical records and experiences to inform decision-making [221, 1068, 50]. By maintaining and utilizing individual memory of past interactions, decisions, and outcomes, the agent can refine its decision-making process over time. This memory serves as a repository of experiences that the agent can draw on when making future decisions. Using this stored knowledge, individual agent is able to refine its decision-making process, learning from previous successes and failures [921, 1051]. For example, in clinical simulation, doctor agents can keep improving treatment performance over time by accumulating experience from both successful and unsuccessful cases [921]. In social behavior simulation, agents can improve their adaptability by engaging in more complex scenarios and utilizing scenario memories to enhance performance [50].

**Shared memory-based learning** In contrast, shared memory-based learning extends this concept by enabling multiple agents to exchange information and insights derived from their respective experiences. Rather than relying solely on individual memory, agents can benefit from the collective knowledge of the group. By sharing data, strategies, and feedback, agents enhance their ability to cooperate and optimize their decisions collaboratively. Shared memory-based learning is particularly valuable in environments where agents need to cooperate, exchange tasks, or work toward common goals [919, 967, 968]. For instance, ProAgent [1069] anticipates teammates' decisions and dynamically adjusts each agent's strategies based on the communication logs between agents, facilitating mutual understanding and improving collaborative planning capability.

**Parameter-based learning.** Beyond memory-based learning in textual form, many MAS employ parameter-based learning, which evolves agents' individual adaptability through post-training techniques. For instance, [1070] discusses the Learning through Communication (LTC) paradigm, where using communication logs between agents are leveraged to construct to generate datasets for training or fine-tuning LLMs. The integration of symbolic and connectionist paradigms within LLM-powered agents enhances both their reasoning and adaptability. More recently, research has increasingly focused on multi-agent (co-)fine-tuning, which improves collaboration and reasoning capabilities through cooperative trajectories. Examples include multi-agent debate fine-tuning [1071] and SiruiS [1072]. Additionally, Sweet-RL [1073] employs reinforcement learning to enhance the critic model within MAS, fostering better collaborative reasoning. However, despite their promising performance, future parameter-based learning paradigms may need to address the balance between agents' general capabilities and their specialization for specific roles within MAS. This hybrid approach allows agents to handle both structured and unstructured data, improving their ability to make decisions in dynamic environments [1074, 1075].

## Chapter 17

# Evaluating Multi-Agent Systems

The transition from single-agent to multi-agent systems, and specifically Large Language Model (LLM)-based systems, requires a paradigm change in the evaluation paradigm. In contrast to single-agent evaluation, in which the immediate concern is performance on a particular task, evaluation of LLM-based multi-agent systems must be understood in terms of inter-agent dynamics as a whole, such as collaborative planning and communication effectiveness. Both task-oriented reasoning and holistic capability evaluation are addressed in this chapter, reflecting the nuance of such evaluations. In greater detail, there are two main areas that we examine for evaluation. First, there is task-solving Multi-Agent Systems (MAS), where we examine benchmarks assessing and enhancing LLM reasoning for coding, knowledge, and mathematical problem-solving tasks. These tests also accentuate the utility of distributed problem solving, achieved through organized workflows, specialisation among agents, iterative improvement, and calls for additional tools. Enhanced reasoning, primarily because of agent-agent decision-making cooperation and multi-round communications, is shown for MAS compared with agent-based individual ones. Following that, there is a general evaluation of MAS abilities, extending beyond one-task-oriented achievement, to agent interactions at a highly advanced level. It involves a move away from one-dimensional measurements into multi-dimensional frameworks for documenting achievements at collaborations, reasoning abilities, system efficiency, and flexibility. We categorize such measurements into collaboration-oriented and competition-oriented measurements and have identified efficiency, decision-making quality, quality of collaboration, and flexibility as primary measure domains. These measurements capture various aspects of agent behavior, including communication effectiveness, resource distribution, and response to dynamic situations.

### 17.1 Benchmarks for Specific Reasoning Tasks

In multi-agent system solving for tasks, much focus has been on leveraging multi-agent coordination for enhancing the reasoning capacity of LLMs. It is most evident in coding, knowledge, and mathematical reasoning benchmarks, where one is interested in examining and building on performance with distributed solving. These benchmarks most typically examine if agents' capability for producing correct code, reasoning on complex knowledge domains, and solving difficult mathematical problems withstanding, with measures such as *pass@k* [1076] or proof ratios for success being prevalent. Much improvement has been exhibited by MAS through structured workflow, domain-specific agent roles, and iterative improvement on state-of-the-art performance. On the contrary, for model and simulation MAS, the case is one with a comparative lack of standardized benchmarks. Rather, research is primarily experimental setups that simulate a variety of social phenomena, with calls from the community for further formalized evaluation frameworks. These multiple benchmark areas are described below, examining the tasks, measures for evaluation, and the core mechanisms through which MAS result in better performance.

**Code Reasoning Benchmark** Measuring the capability of LLMs for code synthesis requires bespoke benchmark suites with a focus on functional correctness. Code synthesis, as compared to natural language synthesis, allows for direct verification through running. Several benchmark suites have been built for this purpose, typically consisting of a collection of programming problems, each described with a natural language problem description and a collection of test cases for automatically ascertaining the synthesized code's correctness. HumanEval [1077], APPS [1078], and MBPP [939] are some popular ones. These benchmark suites predominantly utilize the *pass@k* metric, which computes the percentage at which at least one among the top-*k* generated solutions passes all test cases for a number of problems. The problems covered through these benchmark suites range across a variety of difficulties and programming

abstractions, requiring not only for LLMs and Agents but also for syntactically correct and logically sound code that satisfies the provided test cases. Recent work has explored leveraging Multi-Agent Systems (MAS) for enhancing LLM capability on code reasoning. For instance, MetaGPT [626] is a meta-programming system which embeds human-like Standard Operating Procedures (SOPs) into multi-agent cooperation based on LLM. With multi-agent role assignment with varying domains and adopting assembly line mode, MetaGPT effectively breaks down difficult operations into sub-operations and achieves state-of-the-art performance on HumanEval and MBPP benchmarks. SWE-agent [628] presents a novel Agent-Computer Interface (ACI) which largely enhances a repository-creating, repository-editing, and navigation capability for an agent. The system demonstrates that a well-structured interface tailored for LMs can largely enhance software engineering capability, with state-of-the-art on SWE-bench and HumanEval. AgentCoder [994] is another multi-agent coding system with focus on effective testing and auto-optimization. It is a three-agent system with a programmer, a test designer, and a test executor. The test designer supplies accurate and diverse test cases, and the test executor provides feedback to the programmer for optimization. Such collaborative workflow enhances coding efficiency and outperforms one-agent models and other multi-agent approaches on HumanEval and MBPP datasets. These MAS approaches all point out multi-agent cooperation, organized workflow, and tailored interface as effective solution strategies for enhancing the capability of LLM on code reasoning. DEVAI [781] proposes a set of novel AI development automation benchmarks, which utilize a judge-agent mechanism for judging automatically intermediate development process.

**Knowledge Reasoning Benchmark** To facilitate AI agents effectively acting in and understanding the world, robust knowledge reasoning abilities are essential. Benchmarks for this class assess an agent’s ability to utilize factual knowledge and logical reasoning when answering challenging queries. Commonsense reasoning is tested with benchmarks such as CSQA [1079] and StrategyQA [1080], and scientific knowledge understanding is tested with ScienceQA [1081]. The core challenge for agents is performing multi-step, chain-of-thought reasoning, stepwise logically progressing from input query to output answer. These tests concentrate on assessing how well a specific AI agent can apply a specific body of knowledge, one at a time, and reason out a problem. Recent research has experimented with the use of LLMs on MAS for improving knowledge reasoning task performance, and they have achieved state-of-the-art accuracy. For example, MASTER [1009], a novel multi-agent system, employs a novel recruitment process for agents and communication protocol using the Monte Carlo Tree Search (MCTS) algorithm, and achieves 76% accuracy on HotpotQA [940]. Reflexion [48], a universal framework for bringing reasoning and acting together with language models, improves baseline by 20% on HotpotQA. These strategies demonstrate the potential of multi-agent coordination for knowledge reasoning tasks. Besides, leveraging external tools, e.g., search engines, is also needed for improving knowledge reasoning capacity. Agents may apply these tools for retrieving the latest information and also for fact checking, thus improving the accuracy and dependability of responses. Such integration is particularly helpful on applications such as TriviaQA [1082], for which real-time information access is essential.

**Mathematical Reasoning Benchmark** Math reasoning is a critical skill for AI agents which requires cooperative utilisation of mathematical knowledge, logical deduction, and computational power. Benchmarking tasks for this capability tend to fall into two categories: math problem-solving and computer-aided theorem proving (ATP). Datasets such as SVAMP [942], GSM8K [1083], and MATH [941] challenge agents to solve word problems, asking for exact number answers or formulas. ATP is a harder test, with stricter compliance with formal proof schemata. Tests on datasets like PISA [1084] and miniF2F [1076], which are graded on proof completion, test whether an agent can produce well-formed mathematical proofs. Multi-agent systems (MAS) have been put forward as a potential solution for handling mathematical reasoning problem complexity. Methods such as MACM [1010] include a multi-agent system consisting of Thinker, Judge, and Executor agents tailored for a complex problem, dividing it into smaller sub-problems for computation. The Thinker agent generates new ideas, Judge decides if they are accurate, and Executor conducts necessary computation involving tools such as calculators. Such a modular structure supports iterative refinement and elimination of errors, enhancing problem-solving accuracy. Furthermore, methods such as multi-agent debate [985] include several instances of a language model debating and refocusing iteratively for collective solution improvement, enhancing reasoning as well as factuality accuracy. Such MAS-based systems have achieved notable improvement on benchmarks such as MATH and GSM8K, establishing distributed solving capacity for mathematical problems. Aside from this, reinforcement learning from human feedback (RLHF) and preference learning strategies have been attempted for further enhancing mathematical problem-solving capacity of LLMs. For instance, a multi-turn online iterative direct preference learning framework [1085] has been put forward for training various language models with enriched sets of prompts over GSM8K and MATH datasets. Such a technique includes feedback from interpreters for codes and optimizes preferences at a level of trajectories, with notable improvement in output.

**Societal Simulation Benchmark** Social simulation benchmarks are essential for evaluating multi-agent system performance and realism for simulating human behavior and social interactions based on LLMs. Standardized sets and test cases for evaluating the agents’ ability for interacting, communicating, and evolving within a simulated society are



Table 17.1: MAS Benchmarks: A Systematic Classification of Multi-Agent System Evaluation Frameworks Categorized by Task-Oriented Performance and System-Level Capabilities. This comprehensive collection encompasses both specialized task-solving benchmarks and holistic capability assessments, reflecting the dual nature of MAS evaluation in collaborative problem-solving and inter-agent dynamics.

Category	Focus	Benchmarks	Examples	Representative Metrics
Task-solving	Code Reasoning	APPS [1078], HumanEval [1077], MBPP [939], CodeContest [1087], MTPB [1088], DS-1000 [1089], ODEX [1090], Raconteur [1091]	MetaGPT [626], SWE-agent [628], AgentCoder [994]	Pass@k, Resolved(%)
	Knowledge Reasoning	ARC [1092], HotpotQA [940], CSQA [1079], StrategyQA [1080], BoolQ [1093], OpenBookQA [1094], WinoGrande [1095], HellaSwag [1096], SIQA [1097], PIQA [1098], proScript [1099], ScienceQA [1081], ProOntoQA [1100]	Reflexion [48], MASTER [1009]	Accuracy
	Mathematical Reasoning	MATH [941], GSM8K [1083], SVAMP [942], MultiArith [943], ASDiv [1101], MathQA [1102], AQUA-RAT [1103], MAWPS [1104], DROP [1105], NaturalProofs [1106], PISA [1084], miniF2F [1076], ProofNet [1107]	MACM [1010], Debate [985]	Accuracy, Pass@k
Collaboration	Communication-based Cooperation	InformativeBench [1108], Collab-Overcooked [944], COMMA [1109], LLM-Coordination [926]	iAgents [1108], Two-Player [1110], EAAC [1111]	Task Completion Rate Communication Efficiency
	Planning and Coordination	PARTNR [946], VillagerBench [925], BABYAGI-ARENA [1112], Multiagent Bench [948]	AAS [1113], ResearchTown [1114], GPTSwarm [651]	Planning Success Rate Coordination Efficiency
	Process-oriented	Auto-Arena [947]	Idea [1115]	Process Completion Rate Step Efficiency
Competition	Adversarial Scenarios	BattleAgentBench [920], MAgIC [955], LLMArena [1116], PokerBench [1117], Multiagent Bench [948]	Dilemma [1118], Pok��LLMon [1119]	Win Rate Elo Rating
	Social Deduction	AvalonBench [972], Human Simulacra [1120], Diplomacy [934]	MA-KTO [1121], HLR [1122]	Win Rate Accuracy of Deductions
	Game-Theoretic	Guandan [1123], AgentVerse [1124], ICP [1125]	WarAgent [1126]	Score Win Rate

provided through the benchmarks. An example of one such widely used benchmark is SOTOPIA [1086], employed for evaluating social intelligence in natural language agent-based social intelligence. It is employed for evaluating agents’ ability for conversing, understanding social cues, and building relationships with each other within a virtual society. Another benchmark involves simulating propagation Gender Discrimination and Nuclear Energy [255] topics on social networks. It is employed to evaluate agents’ capabilities in modeling opinion dynamics, information dissemination, and social influence within large-scale social networks. Multiagent Bench [948] further provides two simulation domains—werewolf and bargaining—to assess competitive interactions among diverse agent groups with conflicting goals.

Evaluating capabilities in LLM-based MAS requires specialized approaches that effectively measure the rich interactions between agents. As this field evolves, evaluation methodologies have transitioned from single-dimension metrics to multi-faceted evaluation frameworks that capture the complex skillset required for effective multi-agent interaction. This evolution reflects a growing understanding that agent performance must be assessed across multiple dimensions including collaboration success, reasoning capabilities, and system efficiency.

In recent research, the MAS evaluation can be mainly categorized along three primary dimensions: collaboration-focused benchmarks, competition-focused benchmarks, and adaptive and resilience benchmarks. Within each category, we identify specific metric domains that capture different aspects of agent performance. Current evaluation approaches typically measure efficiency metrics (e.g., task completion rates, resource utilization, time efficiency), decision quality metrics (e.g., action accuracy, strategic soundness, reasoning depth), collaboration quality metrics (e.g., communication effectiveness, coordination efficiency, workload distribution), and adaptability metrics (e.g., response to disruptions, self-correction), which provide a foundation for evaluating multi-agent systems.

**Collaboration-focused Benchmarks.** Collaboration-focused benchmarks have evolved significantly, shifting from basic single-dimensional metrics toward comprehensive frameworks that evaluate complex agent-to-agent communication

and coordination. Initial benchmarks, such as InformativeBench [1108], primarily addressed agent collaboration under conditions of information asymmetry, employing metrics like Precision and IoU to measure decision accuracy in information dissemination tasks. Subsequently, the scope of evaluation expanded, exemplified by Collab-Overcooked [944], which introduced nuanced process-oriented metrics such as Trajectory Efficiency Score (TES) and Incremental Trajectory Efficiency Score (ITES). These metrics assess detailed aspects of coordination, revealing significant shortcomings in agents' proactive planning and adaptive capabilities despite their strong task comprehension.

Further expanding the evaluation scope, COMMA [1109] and LLM-Coordination [926] emphasized communication effectiveness and strategic synchronization, employing diverse environments and extensive metrics including Success Rate, Average Mistakes, and Environment Comprehension Accuracy. These benchmarks collectively illustrate an emerging trend toward capturing deeper aspects of collaborative behaviors and strategic consistency.

Other benchmarks, such as PARTNR [946], VillagerBench [925], and BabyAGI [1112], further addressed gaps in existing evaluations by focusing explicitly on reasoning, planning, and task decomposition. These benchmarks highlighted the need for comprehensive assessment of agents' ability to engage in complex, socially embedded tasks, considering metrics like Percent Completion, Balanced Agent Utilization, and agent contribution rates. AgentBench [706], VisualAgentBench [928], and Auto-Arena [947] further standardized multi-agent evaluations, automating assessment across various domains and demonstrating substantial performance disparities between closed-source and open-source LLMs. These observations underscored critical challenges in developing universally effective collaboration frameworks.

In summary, collaboration-focused benchmarks collectively reflect an ongoing shift toward comprehensive, nuanced evaluations that encompass communication efficiency, adaptive strategy, and fine-grained agent coordination, addressing earlier limitations focused solely on outcome-based performance.

**Competition-focused Benchmarks.** Competition-focused benchmarks evaluate agents' strategic capabilities and adversarial interactions, highlighting specific deficiencies in Theory of Mind and opponent modeling. Early benchmarks such as BattleAgentBench [920] and MAgIC [955] initiated the focus on mixed cooperative-competitive environments, uncovering critical weaknesses in high-order strategic reasoning among LLM agents. These benchmarks employed comprehensive competitive metrics such as Forward Distance, Judgment Accuracy, and Rationality scores, identifying that while advanced LLMs performed adequately in simpler scenarios, significant limitations persisted under complex adversarial conditions.

Building upon these insights, subsequent benchmarks like Human Simulacra [1120], LLMArena [1116], and PokerBench [1117] further refined competitive evaluation by incorporating human-like reasoning metrics and more robust strategic measures (e.g., Response Similarity Score, Elo Scores, and Action Accuracy). These evaluations consistently demonstrated shortcomings in opponent prediction, risk assessment, and adaptive strategic planning, despite high task comprehension.

Social deduction and deception-based benchmarks, notably AvalonBench [972] and Diplomacy [934], further revealed fundamental gaps in agents' abilities to interpret hidden information and manage complex social dynamics. Metrics like Assassination Accuracy, Deduction Accuracy, and Win Rates emphasized that even sophisticated LLMs fail to replicate human-level reasoning in adversarial negotiation and hidden-information games.

Additional game-theoretic evaluations, including Guandan [1123], AgentVerse [1124], MultiAgentBench [948], and ICP [1125], introduced scenarios requiring strategic cooperation under incomplete information. These benchmarks reinforced previous findings on the necessity of enhanced Theory of Mind and predictive modeling capabilities. Multi-AgentBench [948] also introduces the KPI and coordination score to evaluate the competition of agents. Collectively, competition-focused benchmarks highlight persistent strategic and reasoning limitations among LLM-based agents, underscoring the ongoing need to address critical gaps in adversarial modeling and strategic planning despite advancements in general reasoning and task execution capabilities.

**Adaptive and Resilience Benchmarks** adaptive and resilient multi-agent system benchmarks tackle two interconnected capabilities together: adaptability—the ability of the agents to act dynamically in altering, unexpected environmental conditions by modifying their behavior and strategy. Resilience, or the ability of the system to endure, alleviate, and rapidly recover from disruptions, faults, or hostile intervention. In adaptability, as mentioned in AdaSociety [1127], the dynamic interplay between social relationships and physical environments demands that agents engage in continuous learning, and strike a balance between environment discovery and social network construction. Despite significant advancements in current multi-agent decision-making frameworks, these environments fall short in introducing new challenges in various physical contexts and changing social interdependencies. Therefore, AdaSociety introduces an environment in which physical states, tasks, and social relationships among agents continuously evolve, thereby capturing the adaptability of agents as they respond to expanding task complexity and shifting resource constraints.



Moreover, current benchmarks may oversimplify the challenges of real-world automation with limited disruption modeling and simplified dependencies of process [945], resulting in insufficient evaluation of planning capabilities and adaptability. Thus, REALM-Bench [945], on the other hand, defines adaptation through real-world-inspired planning problems, which emphasizes metrics such as real-time re-planning efficiency, coordination scalability under increasing complexity, and the stability of performance outcomes despite dynamic interdependencies or disruptive events. Conversely, resilience benchmarks [1128] systematically introduce faults or errors into individual agents to assess overall system robustness.

## 17.2 Challenge and Future Work

While various MAS evaluation benchmarks have been developed in recent years, challenges and limitations continue to exist with regard to the standardization of evaluation across different MAS tasks and scenarios, and the ability to evaluate scalability and diversity in MASs. Future research must address these challenges, in order to develop the comprehensive field of MAS evaluation.

Below are some challenges and future directions in LLM Multi-agent evaluation:

1. Multi-Agent System has demonstrated superior performance in solving complex tasks, when compared with single agent frameworks. But compared with single agent system, MAS also requires more computations and brings additional costs. Therefore, there has a urgent challenge that we need to handle: when we need to invoke MAS framework? For many simple user instructions, we may only require LLM or single agent system to accomplish. And only complex user instructions could require MAS frameworks. Hence, in the future, how to design the task router mechanism to detect which scenario require MAS or not is fundamental but also a important issue.
2. Multi-agent system is a high-level framework, built upon multiple AI agent based on the foundation models. Therefore, just like back propagation, the optimization of MAS framework will also affect each part (i.e., foundation model, AI Agent and Multi-agent collaboration).
3. Existing MAS frameworks usually design multiple agents with homogeneous traits, such as all being language-based agents. But when connecting MAS to real-world scenarios, it usually involves different kinds of AI agents. For example, we may need to bridge the connections between language-based agent, digital agent and robotic agents. However, these agents adopt various settings, from the inputs to the outputs. How to establish the connection between these agent is still a open problem that need to be handle in the future.